Bertoni, Augusto César

WiWater: Sistema inalámbrico y sin baterías de medición localizada de consumo de agua

Tesis para la obtención del título de grado de Ingeniero Eléctrico Electrónico

Director: Germena, Eduardo Daniel

Ducloux, José María,

WiWater

Sistema inalámbrico y sin baterías
de medición localizada
de consumo de agua

Universidad Católica de Córdoba

Augusto César Bertoni

Noviembre de 2015

Resumen

Este *Trabajo Final* consiste en el estudio, diseño y desarrollo de un sistema integral de medición localizada de consumo de agua para los hogares, cuya finalidad es ayudar, guiar y recomendar al consumidor en el proceso de cambio de hábito para un consumo más responsable.

Reconociendo la crisis global de agua potable, se analiza el comportamiento humano tratando de encontrar la causa de esta situación. A raíz de lo anterior, desde el plano tecnológico se ofrece una herramienta capaz de incentivar el consumo sustentable y mesurar los avances logrados, garantizando una excelente experiencia de usuario debido a su intuitiva interfaz gráfica, su tecnología inalámbrica y su autonomía energética, significando esto último un mantenimiento nulo de los dispositivos de medición.

Partiendo de la premisa de que *—lo que no se puede medir no se puede mejorar—*, este sistema pone en evidencia la buena voluntad de las generaciones ecológicas, promoviendo el cambio y obteniendo resultados a corto y largo plazo.

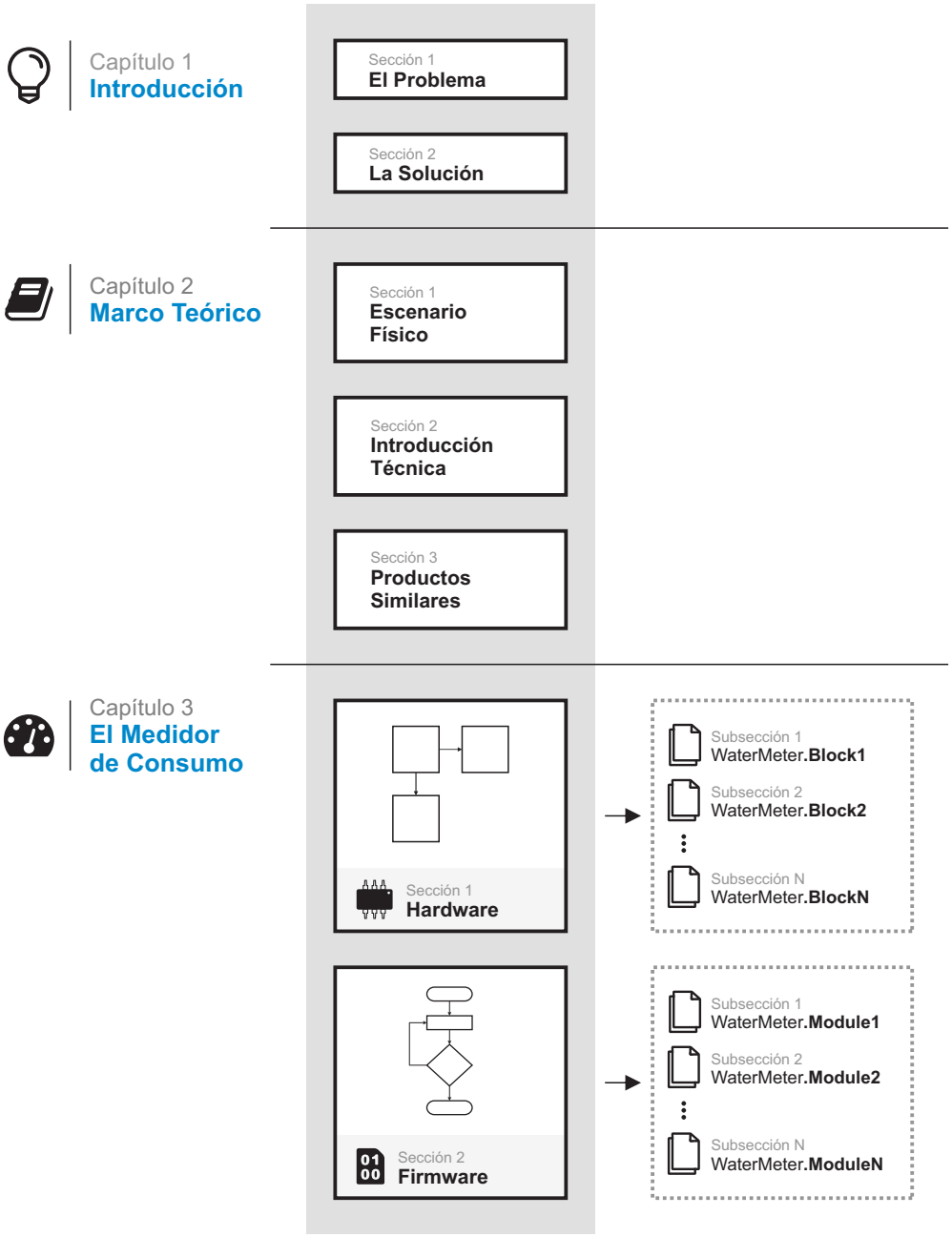
A mi abuela Nelva.

Agradecimientos

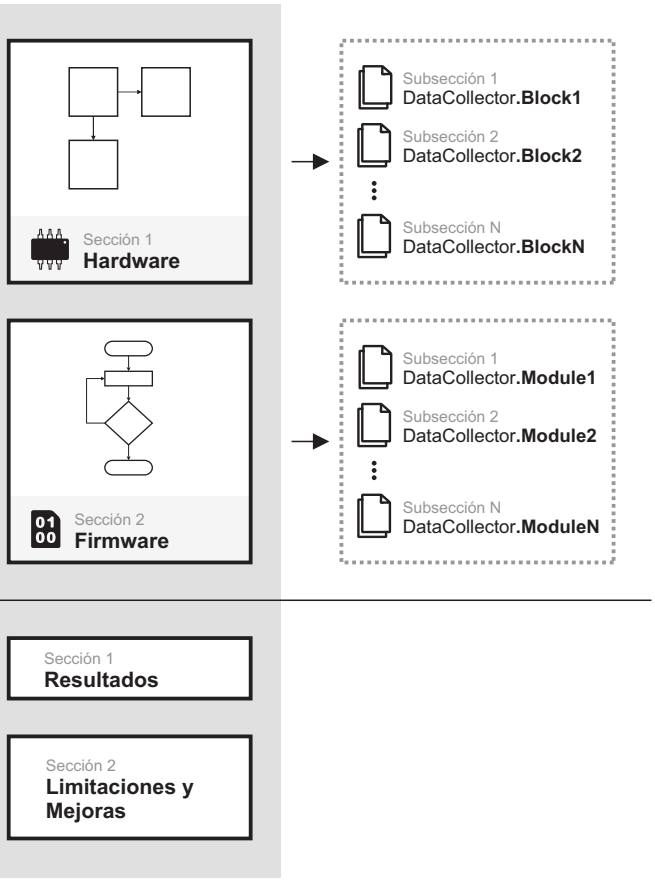
En primer lugar, quisiera agradecer sinceramente a mis tutores de Trabajo Final, Ing. Eduardo Daniel Germena e Ing. José Ducloux, por el apoyo recibido a lo largo de este arduo camino, desde el diseño de los primeros prototipos hasta la confección de la última página de este informe. Sus conocimientos y experiencia han sido de gran utilidad a la hora de tomar importantes decisiones.

Quisiera agradecer también a mi familia, especialmente a mis padres, por permitirme haber llegado hasta aquí. Sin su soporte, esto no hubiera sido posible.

Mapa de contenido



Capitulo 4
El Concentrador de Datos



Capitulo 5
Conclusiones

Índice general

1. Introducción **1**
1.1. El Problema . . . 1
1.1.1. La crisis global de agua potable . . . 1
1.1.2. Cambiar hábitos: tarea difícil . . . 2
1.2. La Solución . . . 3
1.2.1. Educación o castigo . . . 3
1.2.2. Programas educativos . . . 3
1.2.3. Mejores consumidores . . . 4
1.2.4. Desafíos actuales en sistemas de medición . . . 4
1.2.5. La solución propuesta . . . 5

2. Marco Teórico **7**
2.1. Escenario Físico . . . 7
2.1.1. Parámetros físicos: definiciones . . . 7
2.1.2. Modelización de sistemas hidráulicos . . . 9
2.2. Introducción Técnica de la Solución . . . 11
2.2.1. La red . . . 12
2.2.2. El medidor de consumo . . . 12
2.2.3. El concentrador de datos . . . 13
2.3. Productos Similares . . . 14
2.4. Método *Device.Block.Stage* . . . 15

3. El Medidor de Consumo **17**
3.1. *Hardware* . . . 17
3.1.1. Transductor hidroeléctrico . . . 19
3.1.2. Sensor de campo magnético . . . 23
3.1.3. Acondicionamiento de potencia . . . 28
3.1.4. MCU—microcontrolador . . . 43
3.1.5. *Transceiver* y circuito de adaptación de antena . . . 48
3.1.6. LED para *debugging* . . . 51
3.2. *Firmware* . . . 52
3.2.1. Inicialización de BSP . . . 52
3.2.2. El protocolo de la red inalámbrica: SimpliciTI . . . 52
3.2.3. Supervisión de voltaje . . . 57
3.2.4. Enlace con *Access-Point* . . . 58
3.2.5. Modo *sleep* en MCU . . . 58
3.2.6. Lectura y tratamiento de pulsos externos . . . 58
3.2.7. Almacenamiento no volátil de caudal relativo . . . 59

4. El Concentrador de Datos **60**
4.1. *Hardware* 60
4.1.1. Interfaz USB 60
4.1.2. Base de datos y servidor Web 63
4.2. *Firmware* 65
4.2.1. Inicialización de la red 65
4.2.2. Recepción y procesamiento de paquetes 65
4.2.3. Raspbian: RaspberryPi + Debian Linux 67
4.2.4. Introducción a MongoDB 68
4.2.5. Post-procesamiento con Python 70
4.2.6. Meteor: aplicación Web de tiempo real 74

5. Conclusiones **83**
5.1. Resultados 83
5.2. Limitaciones y mejoras 84
5.3. Metodología de trabajo empleada 84

A. Macros de SimpliciTI **86**

Capítulo 1

Introducción

1.1. El Problema

El consumo de agua potable en los hogares no parece ser una preocupación primaria en la actualidad como sí lo está siendo por ejemplo el consumo de energía eléctrica o gas natural. Si bien el agua puede ser considerada un recurso renovable debido a que su ciclo natural de regeneración es más frecuente que su extracción/consumo, teniendo en cuenta el crecimiento de la población, la contaminación y destrucción de fuentes naturales de agua limpia y el mal uso que le damos, será de vital importancia tomar conciencia sobre su aprovechamiento.

1.1.1. La crisis global de agua potable

El Papa Francisco, en la Encíclica *Laudato Si'* publicada a principios de 2015 [1], remarca que el agua es un recurso escaso e indispensable y es un derecho fundamental que condiciona el ejercicio de otros derechos humanos. Eso es indudable y supera todo análisis de impacto ambiental de una región.

Según el autor Robert Glennon, en su libro *Inextinguible: La Crisis del Agua en*

América y Qué Hacer Al Respecto, el agua cumple un papel crítico en virtualmente cada segmento de la economía, desde la industria pesada hasta la producción de comida, desde la fabricación de semiconductores hasta el suministro de Internet. Un futuro próspero depende de una segura y confiable disponibilidad de agua. Y no se la tiene [2].

Cuando se trata de agua, más que con cualquier otro recurso, se arroja por la ventana el sentido común. Se consume agua como si ésta no tuviese valor, y de las formas más ridículas que se puedan imaginar [2]. Esta situación preocupa cada vez más a las empresas encargadas de la distribución de agua. Es necesario que éstas, junto a entidades gubernamentales y fundaciones afines, el sector tecnológico y educativo, den el puntapié inicial instalando una campaña intensa y definitiva de concientización sobre el consumo responsable de agua potable.

El artículo *Medición Inteligente de Aguas Urbanas*, publicado en 2013 [3] por profesionales y docentes de la Universidad de Tecnología de Sydney, Australia, conjuntamente con el Centro de Ingeniería e Infraestructura de la Universidad Griffith, Queensland, Australia, revela que el impacto en el cambio climático, las sequías, el crecimiento de la población y la densificación de los centros urbanos, han aumentado el interés de las compañías proveedoras de agua en adoptar estrategias más sustentables en la distribución del recurso.

Sin embargo, son muchos los obstáculos que se podrían presentar al diseñar y proponer este tipo de campañas ya que muchas veces la calidad de vida de las próximas generaciones y el estado de salud del hogar común no tienen lugar en la ecuación de rentabilidad financiera que, al parecer hoy en día, es la que determina el accionar político y en consecuencia el rumbo de las sociedades. En [1] se hace referencia a este tipo de comportamiento cuando se señala que los progresos científicos más extraordinarios, las proezas técnicas más sorprendentes, el crecimiento económico más prodigioso, si no van acompañados por un auténtico progreso social y moral, se vuelven en definitiva contra el hombre. En todo caso debe quedar en pie que la rentabilidad no puede ser el único criterio a tener en cuenta. La protección ambiental no puede asegurarse sólo en base al cálculo financiero de costos y beneficios. El ambiente es uno de esos bienes que los mecanismos del mercado no son capaces de defender o de promover adecuadamente.

1.1.2. Cambiar hábitos: tarea difícil

Toda pretensión de cuidar y mejorar el mundo supone cambios profundos en los estilos de vida, los modelos de producción y de consumo, las estructuras consolidadas de poder que rigen hoy la sociedad. Un desarrollo tecnológico y económico que no deja un mundo mejor y una calidad de vida integralmente superior no puede considerarse progreso [1].

Tanto desde la Santa Sede como desde las universidades más importantes del mundo existe un llamado a la humanidad a cambiar sus hábitos de consumo y producción. La conciencia de la gravedad de la crisis cultural y ecológica necesita traducirse en nuevos hábitos [1]. Pero como se sabe, modificar los hábitos no es tarea sencilla. Dependerá de la efectividad de los programas de concientización y/o de la rigurosidad de las penalizaciones aplicadas a los consumidores que no deseen colaborar. En [1] se menciona que

es un desafío educativo. Sin embargo, esta educación, llamada a crear una «ciudadanía ecológica», a veces se limita a informar y no logra desarrollar hábitos.

1.2. La Solución

1.2.1. Educación o castigo

Existe una fuerte discusión entre distintos sectores preocupados por esta situación, que enfrenta las dos opciones más populares respecto a cómo crear conciencia ecológica: una, educando a los consumidores, y la otra, castigándolos si se comportan fuera de los límites establecidos. En [1] se menciona algo muy interesante: «La existencia de leyes y normas no es suficiente a largo plazo para limitar los malos comportamientos, aun cuando exista un control efectivo. Para que la norma jurídica produzca efectos importantes y duraderos, es necesario que la mayor parte de los miembros de la sociedad la haya aceptado a partir de motivaciones adecuadas, y que reaccione desde una transformación personal». A partir de estos dichos, se quiere rescatar que depende de la voluntad de las personas lograr un cambio de hábito en la manera de consumir. El método que puede prevalecer a largo plazo es educar ecológicamente a los ciudadanos para que éstos puedan transformar progresivamente sus actos de consumo responsable en hábitos; buenos hábitos.

Sin embargo, algunos opinan que incrementar el precio del agua impulsaría a los individuos particulares, granjeros e industriales a examinar cuidadosamente cómo usan el agua, para qué la usan y en qué cantidades. Los economistas afirman que este incremento en las tarifas alentaría a los usuarios a eliminar el uso marginal que se le da al agua y a destinarla a usos más productivos. Tal aumento del bien estimularía el desarrollo de nuevas tecnologías relacionadas al cuidado y extracción del agua [2].

1.2.2. Programas educativos

En la actualidad, ya son varias las iniciativas y los programas orientados a la conservación de agua potable. Por ejemplo, el sistema de aguas de San Antonio, Estados Unidos, patrocina este tipo de programas. En 2007, cuando empeoró la sequía, el alcalde de San Diego, California, hizo un llamado a la población a conservar el agua. Para ello lanzó el programa voluntario *"20-gallon challenge"* ("El desafío de 75 litros"), pidiendo a la población reducir su consumo diario lo más que puedan [2]. Estos programas por lo general funcionan debido a que la población de América es por naturaleza generosa, ayudando a contribuir al bienestar de la comunidad.

Si la educación ecológica no germina como una revelación en los individuos, será necesaria la repetición constante de las campañas de concientización. Tener en cuenta que los comportamientos que son elegidos por las sociedades se transmiten de generación en generación, como buenas costumbres. Sin embargo, no sucede lo mismo con aquellos comportamientos que son impuestos por alertas temporarias o sistemas de penalización.

Es muy noble asumir el deber de cuidar la creación con pequeñas acciones cotidianas, y es maravilloso que la educación sea capaz de motivarlas hasta conformar un estilo de vida. La educación ambiental debería disponernos a dar ese salto hacia el misterio, desde donde una ética ecológica adquiere su sentido más hondo [1].

1.2.3. Mejores consumidores

Para mejorar la situación actual y futura respecto a la crisis del agua en el mundo, se puede expandir el suministro mediante el reciclado de los efluentes municipales y a través de la desalinización de los océanos. Sin embargo, ninguna de estas alternativas es una panacea. Desde el lado del consumidor, se puede promover la conservación. En regiones donde se derrocha, la conservación tiene un gran potencial [2].

La acción política local puede orientarse a la modificación del consumo, al desarrollo de una economía de residuos y de reciclaje [1]. Tener en cuenta que acciones de conservación como las de este tipo, son acciones que muestran resultados a largo plazo, por lo que muchas veces no son atractivas para el sector político. Como se menciona en [1], «los resultados requieren mucho tiempo, y suponen costos inmediatos con efectos que no podrán ser mostrados dentro del actual período de gobierno». Los esfuerzos para un uso sostenible de los recursos naturales no son un gasto inútil, sino una inversión que podrá ofrecer otros beneficios económicos a medio plazo . Un camino de desarrollo productivo más creativo y mejor orientado podría corregir el hecho de que haya una inversión tecnológica excesiva para el consumo y poca para resolver problemas pendientes de la humanidad. Decía Benedicto XVI que «es necesario que las sociedades tecnológicamente avanzadas estén dispuestas a favorecer comportamientos caracterizados por la sobriedad, disminuyendo el propio consumo de energía y mejorando las condiciones de su uso» [1].

Mucha gente desconoce lo que consume de agua. Una encuesta en Estados Unidos en el año 2007 preguntó a la población qué opinaba respecto a su propio consumo: si estaba por encima, en el promedio o por debajo de él. Un remarcable 87 % creía que su consumo de agua estaba en el promedio o por debajo de él . Evidentemente hay un factor humano en juego: nuestra percepción sobre nosotros mismos se aleja dramáticamente de la realidad [2].

1.2.4. Desafíos actuales en sistemas de medición

Para promover la responsabilidad individual (teniendo en cuenta que el contexto, la cultura y los hábitos de consumo también son importantes), los consumidores particulares necesitan tener acceso a información instantánea, relevante y comprensible que pueda asistirlos en el proceso de decisión diaria acerca de cómo consumir el recurso [3].

Actualmente, el desarrollo de medidores inteligentes está orientado a aumentar la cantidad de datos tales como el tiempo de uso y el tipo de uso (ducha, lavamanos, jardín, etc.), como así también la posibilidad de que la tecnología pueda reemplazar las lecturas manuales de los consumos. Sin embargo, el desarrollo tecnológico asociado al uso, medición y cuidado del agua generalmente es menor y está atrasado en comparación al observado en servicios como la energía eléctrica [3].

Cuestiones que han recibido poca atención hasta el momento deben ser atacadas ahora mismo, como es la importancia de la información en tiempo real en la relación con el cliente y la gestión de la demanda. Mientras que los recientes desarrollos en computación y telefonía móvil han mejorado esta situación, la utilidad de esta información se ve limitada en última instancia por la frecuencia y resolución de los datos generados en la fuente, en este caso, los medidores de agua. La provisión de información más detallada acerca del uso del agua a los consumidores está ganando tracción, guiando los desarrollos en el sector energía [3].

Mientras que estas redes de transmisión están disponibles, cuestiones relacionadas a la confiabilidad de las redes inalámbricas, *black spots*, fuentes de energía y vida de la batería, daños por parte de los usuarios, posibilidad de sumergirse y conexiones cruzadas, para nombrar algunas, podrían suponer problemas de fiabilidad [3].

1.2.5. La solución propuesta

Considerando la importancia que tiene la educación para modificar a conciencia el comportamiento relacionado al consumo de los recursos, en este caso el agua, y que este método parece ser el más potente para generar hábitos de consumo sustentable, se diseñó y desarrolló un sistema de base tecnológica capaz de ayudar, guiar y recomendar al consumidor en el proceso de *cambio de hábito*. Para cumplir su objetivo, el sistema mide el consumo de agua de manera localizada, esto es, con medidores en todos los sectores y dispositivos del hogar que consuman agua. El sistema desarrollado ofrece datos que para el consumidor puede ser revelador, y que de ninguna otra forma podría llegar a ver esa realidad.

Lo que no se puede medir, no se puede mejorar

El sistema desarrollado es capaz de registrar el consumo en los grifos del baño y de la cocina, el lavaplatos y lavarropas, el riego de los espacios verdes, la pileta de natación, entre otros. Como se observa en la figura 1.1, se instalan tantos medidores como puntos de consumo existan en el hogar. Además de medir los consumos, el sistema es capaz de almacenarlos y mostrarlos en tiempo real por medio de una aplicación web en cualquier *smartphone* o *tablet*. A su vez, el post-procesameinto de la información por parte del sistema permite realizar comparaciones con consumos anteriores brindando alertas y sugerencias para mejorar día a día la eficiencia en el consumo de agua.

Para que el sistema sea realmente de ayuda al consumidor, se lo ha diseñado para que no necesite mantenimiento. Gracias a su funcionamiento autónomo no posee baterías para funcionar, lo que anula completamente mantenimientos tanto preventivos como correctivos. Esta importante característica permite que los medidores puedan recolectar datos de manera continua y prolongada a lo largo del tiempo, logrando que la información obtenida (gracias al post-procesamiento) sea correcta y de utilidad.

Las comunicaciones inalámbricas de muy bajo consumo son relativamente recientes, por lo que sus aplicaciones, aunque existen muchas, no han cubierto todas las necesidades de la sociedad. Es por ello que el sistema propuesto a nivel tecnológico es novedoso pudiendo ser el único en el mercado local e internacional.

Si bien la consciencia actual sobre el consumo responsable de agua potable no es un factor que demande el producto, será el factor económico y/o el factor escasez el que motive la demanda.

A modo de ejemplo, propietarios de casas de fin de semana situadas en las sierras de Córdoba, sufren escasez de agua debido a que en ciertas zonas las precipitaciones son muy pobres. En esta situación, lo económico queda de lado siendo de gran importancia cómo administrar el recurso para que alcance a lo largo de la estadía. Cuando la disponibilidad de agua potable comience a disminuir en las ciudades, aumentarán sus costos y sus multas por consumos excesivos. Cuando esto ocurra, tendrá un mayor peso la idea de medir el recurso.

Figura 1.1: *WiWater* en un hogar convencional.

Analizando su inexistencia en el mercado local e internacional y su potencialidad como producto tanto en el presente como en el futuro, es una gran oportunidad comenzar *hoy* a desarrollar, probar y producir el sistema propuesto.

El mismo sistema se propone como herramienta educativa con el propósito de educar tanto a niños como adultos sobre consumos sustentables mediante la experiencia de la comparación (por ejemplo, ducharse 10 minutos significa consumir aproximadamente 120 litros de agua, lo que equivale al agua que algunas familias del norte argentino consumen en pocas semanas; cerrar el grifo durante el cepillado de los dientes significa reducir un 30 % el consumo mensual en el hogar). Como estos ejemplos, pueden surgir muchos otros, pudiendo impresionar y como consecuencia educar.

Capítulo 2

Marco Teórico

2.1. Escenario Físico

En la figura 2.1 se observa una implementación típica del sistema propuesto, compuesto de un tanque de agua y tres medidores independientes justo antes de sus respectivos puntos de consumo (lavarropas, punto de riego y grifo para higiene personal).

Antes de avanzar con aspectos específicos del sistema, es necesario analizar y comprender el escenario físico en el que se implementa tal solución. A continuación se describen las variables más comunes involucradas en cualquier sistema hidráulico. Además, partiendo de la configuración de la figura 2.1, se modeliza matemáticamente un sistema hidráulico típico con el fin de dar soporte teórico al sistema presentado.

2.1.1. Parámetros físicos: definiciones

Caudal

En dinámica de fluidos, caudal es la cantidad de fluido que circula a través de una sección del ducto (tubería, cañería, oleoducto, río, canal, etc.) por unidad de tiempo. Normalmente se identifica con el flujo volumétrico o volumen que pasa por un área dada en la unidad de tiempo.

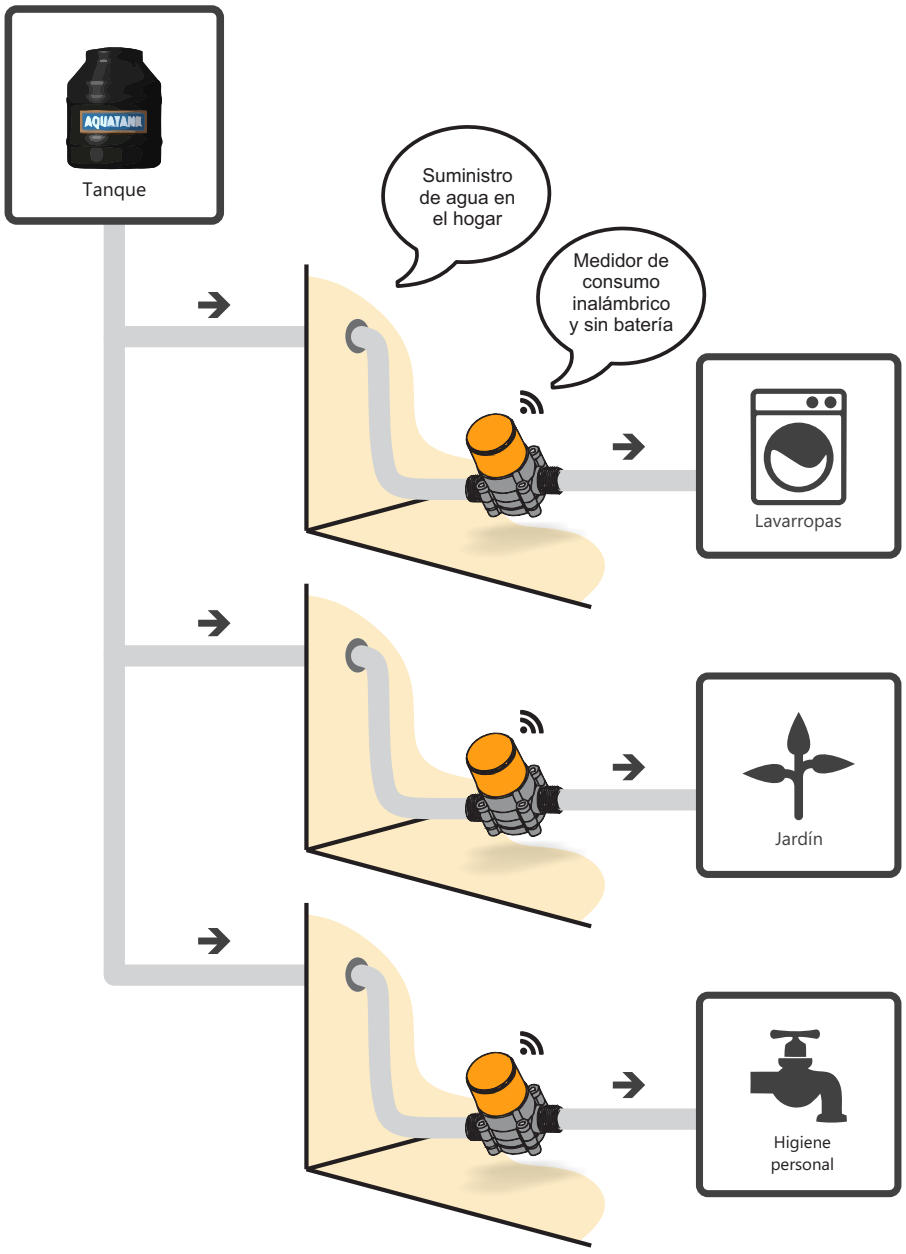


Figura 2.1: Medidores instalados en red de agua dentro del hogar.

En el caso de que el flujo sea normal a la superficie o sección considerada, de área A, entre el caudal y la velocidad promedio del fluido existe la relación:

$$Q = A\bar{v} \tag{2.1}$$

donde:

- Q : caudal, flujo o *flow-rate* $[m^3/s]$
- A : área $[m^2]$
- $\bar{v}$: velocidad promedio $[m/s]$

Presión hidrostática

Un fluido pesa y ejerce presión sobre las paredes del fondo del recipiente que lo contiene y sobre la superficie de cualquier objeto sumergido en él. Esta presión, llamada presión hidrostática, provoca, en fluidos en reposo, una fuerza perpendicular a las paredes del recipiente o a la superficie del objeto sumergido sin importar la orientación que adopten las caras. Si el líquido fluyera, las fuerzas resultantes de las presiones ya no serían necesariamente perpendiculares a las superficies. Esta presión depende de la densidad del líquido en cuestión y de la altura del líquido con referencia del punto del que se mida.

Se calcula mediante la siguiente expresión:

$$P = \rho g h + P_0 \tag{2.2}$$

donde:

- P : presión hidrostática $[pascales]$
- ρ : densidad del líquido $[kg/m^3]$
- g : aceleración de la gravedad $[m/s^2]$
- h : altura del fluido $[m]$. Un líquido en equilibrio ejerce fuerzas perpendiculares sobre cualquier superficie sumergida en su interior
- P_0 : presión atmosférica $[pascales]$

Notar que si se consideran invariables ρ (se supone siempre el mismo fluido), g y P_0 (se mantienen las condiciones físicas del entorno), se puede afirmar que la presión hidrostática P depende solamente de la altura h del líquido:

$$P \propto h$$

2.1.2. Modelización de sistemas hidráulicos

Un sistema hidráulico puede ser interpretado como un sistema eléctrico debido a ciertas similitudes en el comportamiento de ambos. El modelo eléctrico se crea a partir de equivalencias matemáticas. El propósito de crear equivalentes eléctricos para sistemas de distintas naturalezas es predecir su comportamiento y de esta forma manipular sus variables para obtener resultados acordes a lo esperado.

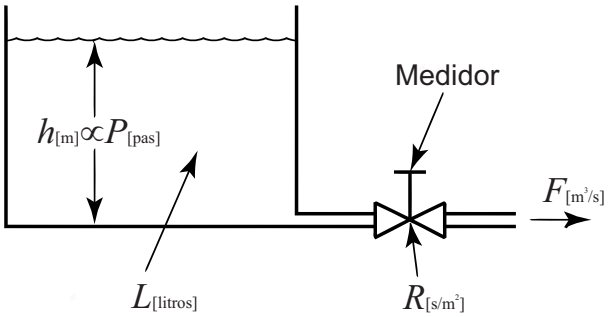


Figura 2.2: Sistema hidráulico a modelizar.

En la figura 2.2 se observa el sistema hidráulico a modelizar. A diferencia del sistema de la figura 2.1, se expone un sólo medidor con el fin de simplificar el análisis.

Existen dos formas distintas de relacionar la columna de agua h y el flujo F: una relación es lineal si el flujo es *laminar*; la otra es no-lineal si el flujo es *turbulento* [4]. Para un flujo laminar la relación es:

$$F = kh \tag{2.3}$$

Entonces, a partir de la ecuación anterior el flujo queda:

$$\begin{aligned} R_{lam} &= dh/dF \\ R_{lam} &= h/F \\ F &= h/R_{lam} \end{aligned} \tag{2.4}$$

En cambio, para un flujo turbulento la relación entre F y h es:

$$F = k\sqrt{h} \tag{2.5}$$

Entonces, a partir de la ecuación anterior el flujo queda:

$$\begin{aligned} R_{tur} &= dh/dF \\ R_{tur} &= 2h/F \\ F &= 2h/R_{tur} \end{aligned} \tag{2.6}$$

Equivalencias entre sistema hidráulico simple y sistema eléctrico

Si se considera lineal el flujo del sistema (en caso de no serlo se puede linealizar por tramos), de acuerdo a la ecuación 2.4 es acertado ver al tanque de la figura 2.3 como un capacitor en donde la altura de la columna de agua h es el voltaje V del capacitor y la cantidad de agua L en *litros* es la cantidad de cargas Q en *coulombs*:

$$Q = CV \rightarrow L = C_{tanque}h$$

donde C_{tanque} queda definido con la unidad $litros/m$, y se podría interpretar como la capacidad volumétrica del tanque de agua, lo que guarda cierta coherencia al relacionarla al valor de capacitancia en $faradios$ que físicamente posee un capacitor. A continuación, se listan las variables involucradas en el sistema hidráulico y sus equivalencias para un sistema eléctrico.

- Presión $P[pascales] \propto$ Columna de agua $h[m] \rightarrow$ Potencial eléctrico $V[voltios]$
- Resistencia al flujo $R[s/m^2] \rightarrow$ Resistencia eléctrica $R[ohms]$
- Flujo de agua $F[m^3/s] \rightarrow$ Corriente eléctrica $I[amperes]$
- Cantidad de agua $L[litros] \rightarrow$ Carga eléctrica $Q[coulombs]$

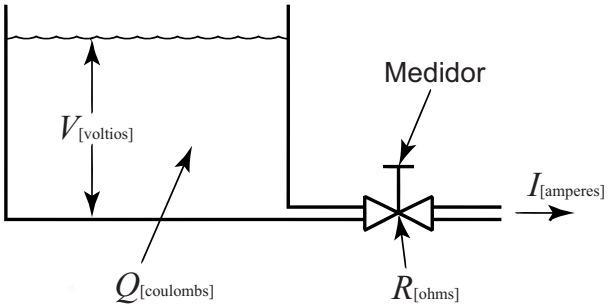


Figura 2.3: Sistema hidráulico con variables eléctricas.

Si se supone que el tanque tiene un sistema de ingreso de agua por válvula flotante que permite mantener h siempre al mismo nivel, se puede considerar al dispositivo tanque/capacitor como a uno tanque/fuente. De esta manera, y para facilitar el uso del modelo eléctrico, la cantidad de agua (las cargas) nunca se agotaría, manteniéndose constante la columna de agua (el voltaje). El circuito eléctrico queda como el de la figura 2.4.

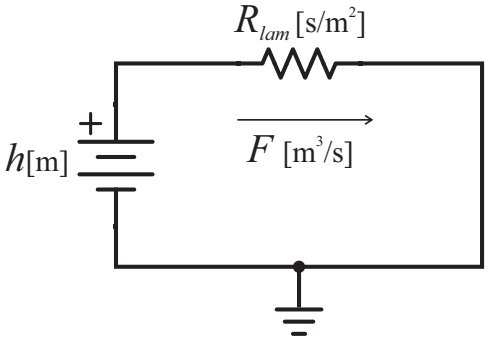


Figura 2.4: Modelo eléctrico equivalente.

Si bien la unidad de la columna de agua es metros, no se debe olvidar de que se supone siempre que el fluido es agua. Notar por la ecuación 2.2 que si se cambia el fluido por alguno de otra densidad, mercurio por ejemplo, la presión de la columna estará dada en metros de Hg o mm(Hg) como, por ejemplo, suele expresarse la presión atmosférica.

2.2. Introducción Técnica de la Solución

Como se mencionó en la sección 1.2.5, la solución propuesta consiste en una red de medidores de consumo de agua, que no necesitan baterías para funcionar y que transmiten los datos de manera inalámbrica a una central que captura, almacena y procesa la información recolectada, permitiendo su visualización a través de cualquier navegador Web.

2.2.1. La red

La red consiste en un grupo de medidores y un concentrador de datos. Como se ve en la figura 2.5, la topología de red es de tipo estrella. El número máximo de dispositivos de medición permitidos por la red es 40. Como se verá más adelante, el protocolo es capaz de reconocer y agregar nuevos nodos.

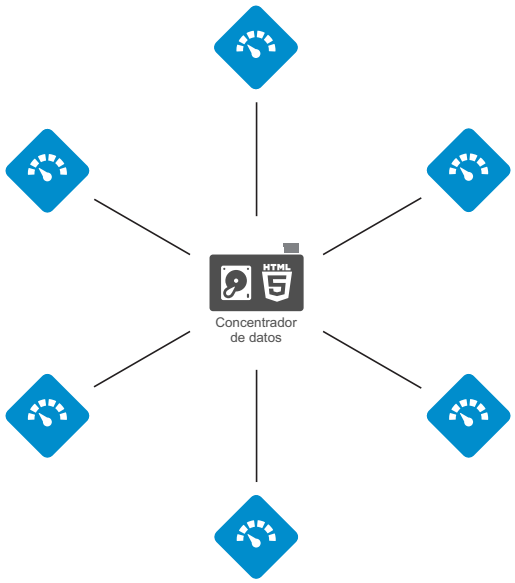


Figura 2.5: Red de sensores y concentrador en topología estrella.

2.2.2. El medidor de consumo

Como se resume en el esquema de la figura 2.6, el dispositivo medidor consta de dos bloques principales: la extracción de energía del flujo de agua y la medición del mismo.

Extracción de energía

Para extraer la energía necesaria para que el medidor funcione se utiliza un transductor hidroeléctrico que convierte la energía cinética del flujo de agua en energía eléctrica. La misma debe acondicionarse con electrónica diseñada especialmente para este tipo de escenarios para que en etapas posteriores sea aprovechada por la electrónica encargada de medir, procesar y transmitir los datos de interés.

Medición, procesamiento y transmisión

Para medir el caudal que pasa a través del generador hidroeléctrico es necesario el uso de algún tipo de sensor que detecte las revoluciones del rotor. El microcontrolador se encarga de leer la salida de este sensor que, por lo general para este tipo de aplicaciones se trata de un sensor de efecto Hall (se profundiza en la sección 3.1.2). Además de capturar los pulsos provenientes del sensor de rotación, el microcontrolador arma los paquetes antes de ser transmitidos y establece la comunicación inalámbrica con el concentrador de datos.

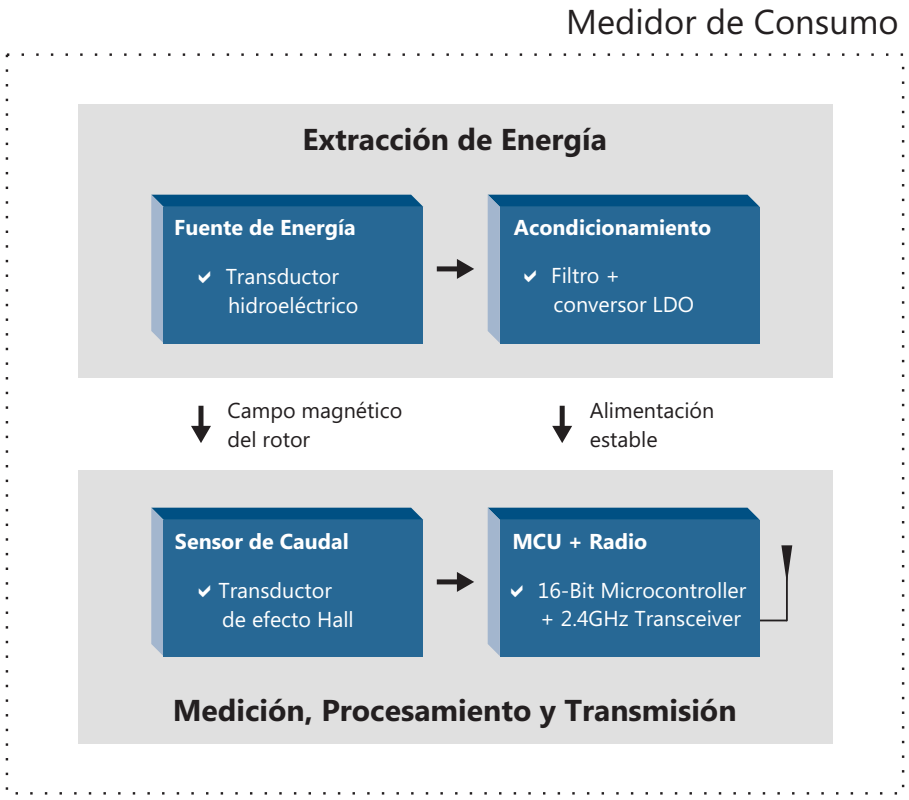


Figura 2.6: Esquema generalizado del medidor.

2.2.3. El concentrador de datos

Como se resume en la figura 2.7, la central que recolecta los paquetes, o concentrador de datos, está compuesta de dos grandes bloques: por un lado la recepción, procesamiento y almacenamiento de los paquetes, y por el otro la visualización de los mismos.

Recepción, procesamiento y almacenamiento

El microcontrolador se encarga de gestionar el *transceiver* para poder recibir los paquetes provenientes de los distintos sensores de flujo. Estos paquetes se envían al administrador de la base de datos que los procesa principalmente para encontrar errores y calcular caudales relativos para luego guardarlos en colecciones correspondientes dentro de la base de datos.

Interfaz gráfica y acceso inalámbrico

Este bloque es sumamente importante y muchas veces subestimado por parte de los desarrolladores. Se trata de la parte que permite establecer una comunicación entre el sistema y el usuario. Si esta etapa no cumple su papel de manera satisfactoria está garantizado el fracaso del producto.

El sistema ofrece la información procesada a través de una dirección del protocolo de Internet (IP, *Internet Protocol*) utilizando como medio físico un enlace inalámbrico WiFi. La aplicación que permite recorrer las funcionalidades del producto está construida en HTML5, lo que permite su visualización en cualquier navegador web, hoy presentes en la mayoría de los dispositivos *tablet* o *smartphone*, como así también en cualquier computadora personal o de escritorio. El funcionamiento consiste básicamente en consultas a

la base de datos por parte de la aplicación Web dependiendo de lo que haya elegido el usuario o lo que el sistema por decisión propia quiera informar.

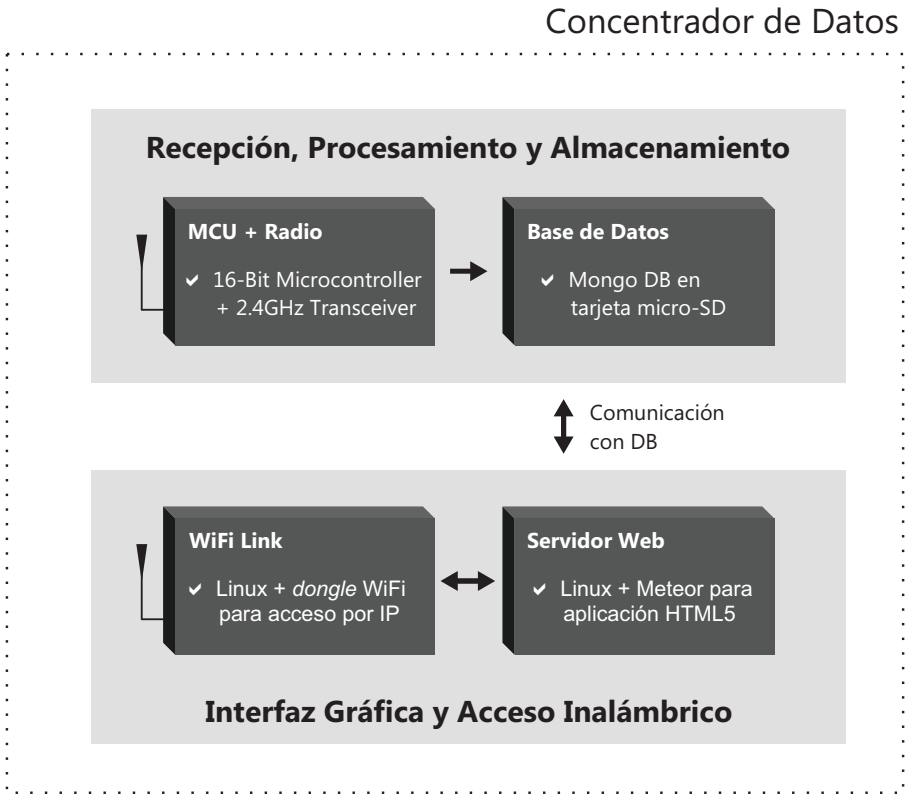


Figura 2.7: Esquema generalizado del concentrador de datos.

2.3. Productos Similares

Se comercializan medidores electrónicos de caudal para usos en el hogar (figura 2.8) que necesitan baterías para funcionar y *displays* para mostrar los valores sensados. Su capacidad de almacenamiento es limitada y no poseen ningún tipo de post-procesamiento.

(a) Uso en el hogar.

(b) Uso múltiple.

Figura 2.8: Medidores en el mercado, con batería y sin *wireless link*.

En el mercado, tanto local como internacional, no existen redes de sensores de consumo de agua que trabajen en conjunto de manera inalámbrica, con medidores sin baterías, y que analicen los datos recolectados guiando, recomendando y advirtiendo al usuario sobre cómo consumir el recurso agua potable.

2.4. Método *Device.Block.Stage*

Este método de organización consiste en generar documentación dividida en secciones a partir del diagrama de bloques del dispositivo que se quiera documentar. A modo de ejemplo para explicar mejor el método, se propone en la figura 2.9 el diagrama de bloques de un *Dispositivo A*. Todo el *hardware* del Dispositivo A es representado por los tres bloques interconectados de la figura.

Por cada bloque del diagrama se generan los documentos necesarios para describirlo desde su concepto hasta su materialización y testeo. Este método propone cuatro documentos o secciones diferentes que corresponden a cuatro etapas o *stages* del bloque: «*IdeaDesign*», «*PartsOnHand*», «*Assembling*» y «*Testing*».

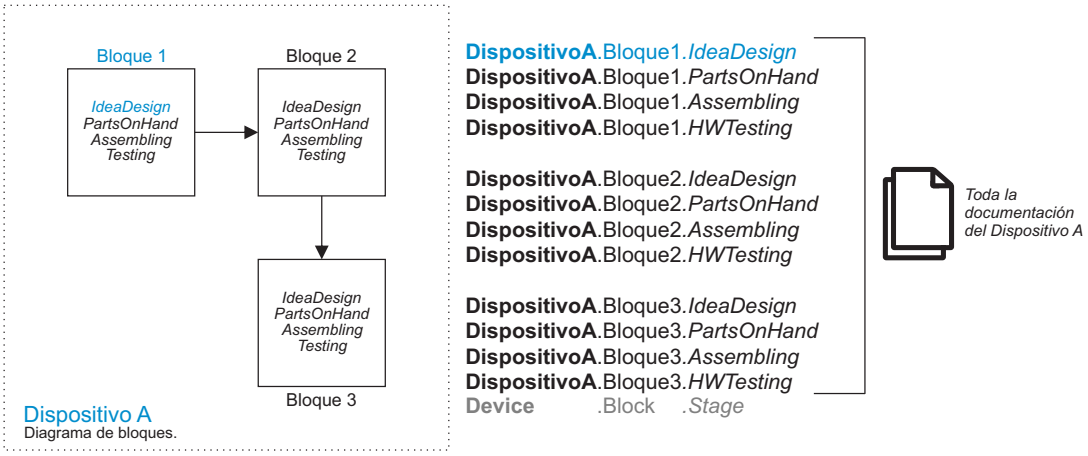


Figura 2.9: Método *Device.Block.Stage*.

IdeaDesign

- Propósito del bloque. Definición.
- Conceptos teóricos.
- Esquemático.

PartsOnHand

- Listado de materiales (BOM, *Bill Of Materials*) por modelo y marca.
- Distribuidor y disponibilidad.
- Costos y envío.

Assembling

- Restricciones de *hardware*.
- *Footpints*: encapsulado de los componentes.
- Diseño de circuito impreso (PCB, *Printed Circuit Board*).
- Fabricación.

Testing

- Curvas y tablas.
- Pruebas, resultados y limitaciones del bloque.

Las secciones se identificarán con el nombre de la etapa (*Stage*), más el del bloque (*Block*), más el del dispositivo correspondiente (*Device*); todos ellos separados por puntos y siguiendo un orden jerárquico. Una sección genérica adoptaría el nombre **Device.Block.Stage**.

Capítulo 3

El Medidor de Consumo

3.1. *Hardware*

El *hardware* correspondiente al medidor se representa con el diagrama de bloques de la figura 3.1. Según el método comentado en la sección 2.4, cada bloque será dividido en cuatro etapas con el fin de abarcar todo el material que se necesita exponer. Se puede ver al diagrama de bloques del *hardware* del medidor como un índice gráfico del contenido de esta sección.

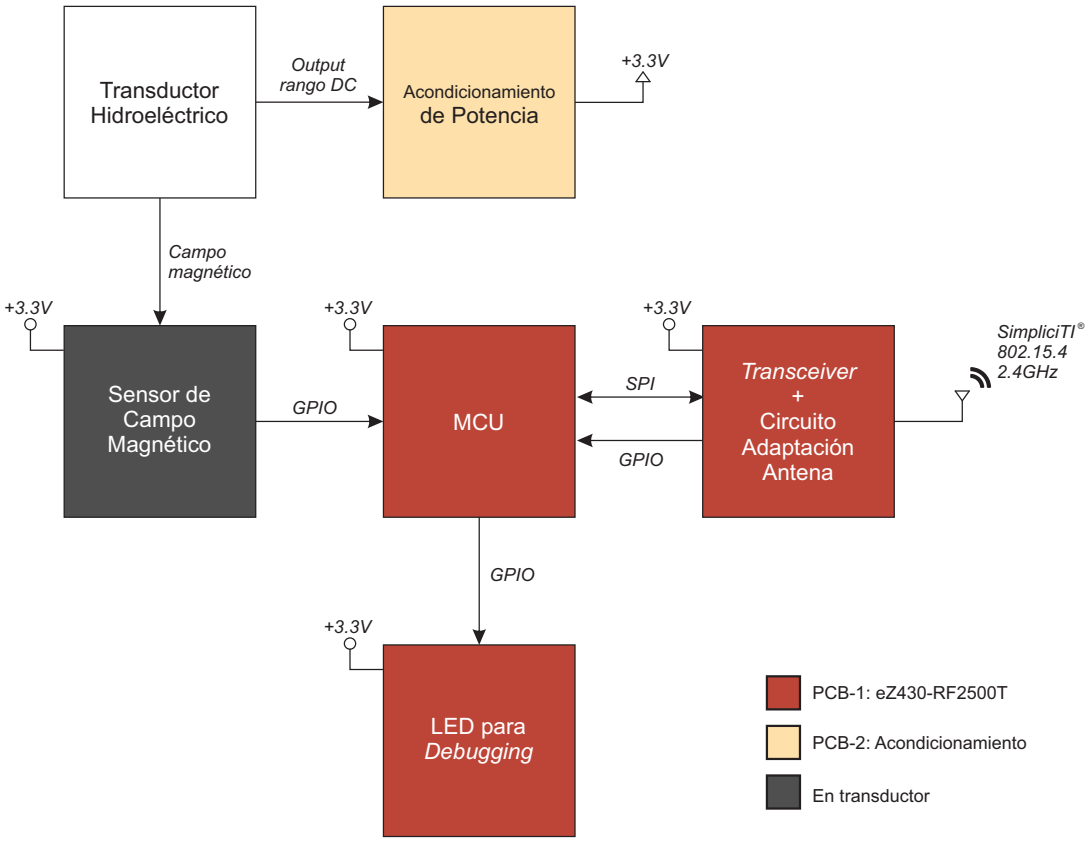


Figura 3.1: Diagrama de bloques del *hardware* del medidor.

3.1.1. Transductor hidroeléctrico

Medidor.Transductor.*IdeaDesign*

Este bloque es uno de los más importantes del producto, ya que permite la ventaja competitiva más fuerte dentro del segmento: «medición sin batería».

El generador convierte la energía cinética del flujo de agua en energía eléctrica. A la hora de elegir una alternativa se pensó en algo seguro para la electrónica instalada para que ésta no sea afectada por el agua. El diagrama de bloques del generador se muestra en la figura 3.2. Sólo el sensor de campo magnético entra en contacto con el agua ya que es montado en cercanía del rotor húmedo.

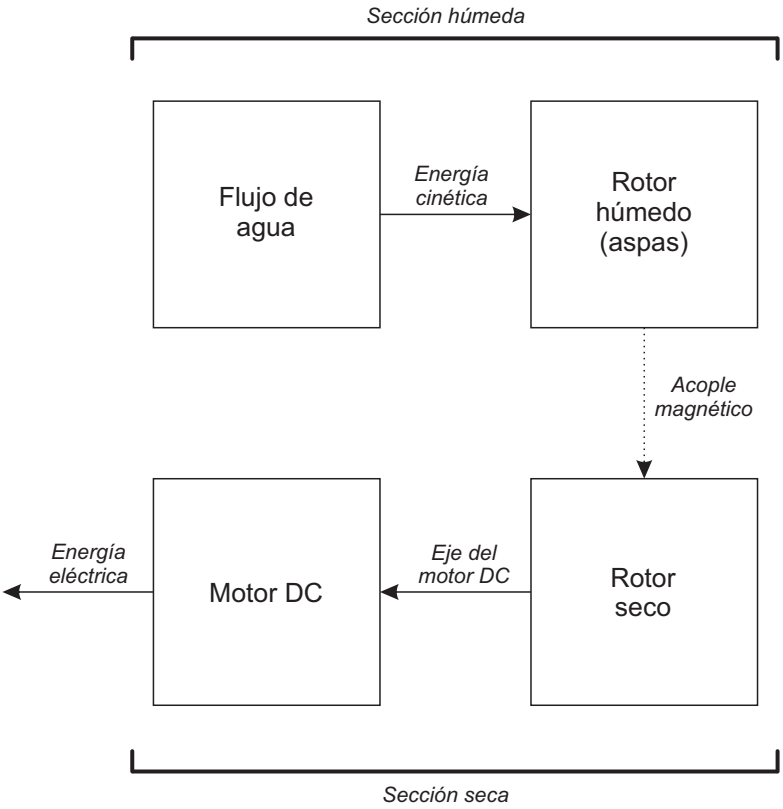


Figura 3.2: Diagrama de bloques del generador.

Medidor.Transductor.*PartsOnHand*

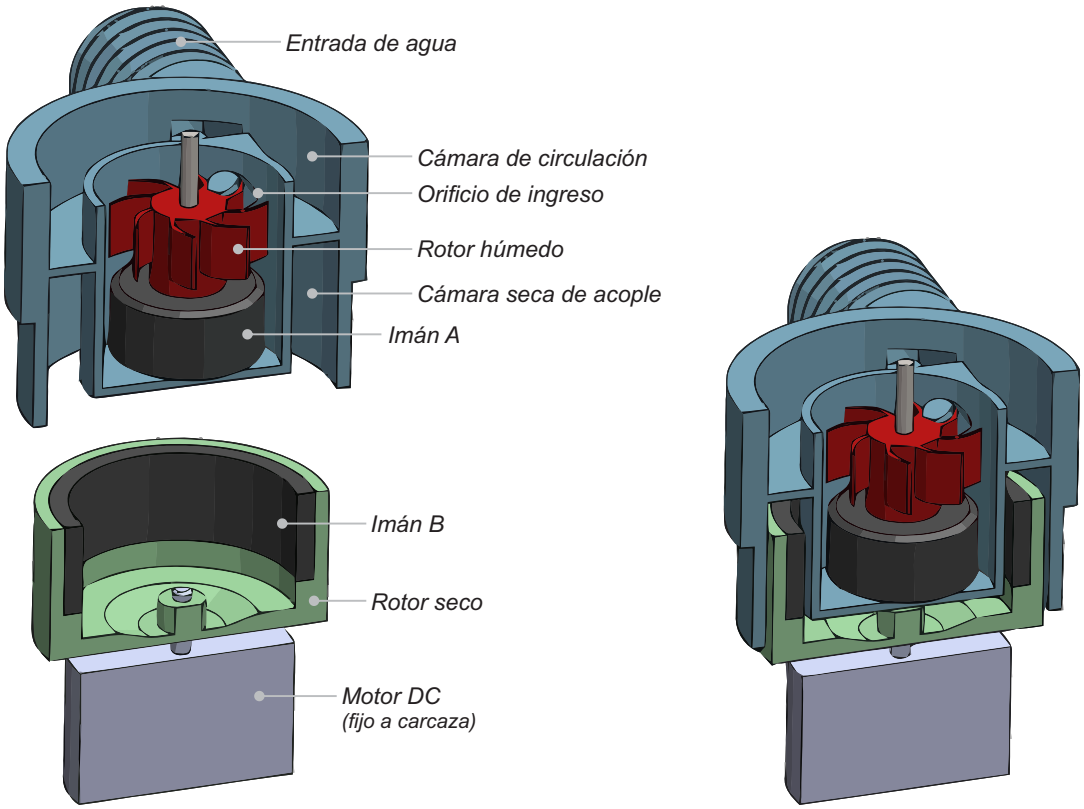
- Fabricante: Desconocido.
- Proveedor: *Aliexpress* [5]
- Origen: China
- Costo: USD 15 c/u (Marzo 2015)
- Envío: Gratis. Demora 45 días.

En la figura 3.3 se ven los dos generadores adquiridos con mangueras flexibles conectadas.

Figura 3.3: Generadores adquiridos en *Aliexpress*.

Medidor.Transductor.*Assembling*

En la figura 3.4a se muestra un modelo en 3D del generador nombrando sus principales partes. El rotor húmedo se acopla magnéticamente con el rotor seco mediante los imanes A y B como puede verse en la figura 3.4b.



(a) Desacoplado. (b) Acoplado.

Figura 3.4: Corte del modelo 3D del generador.

Medidor.Transductor.*Testing*

La función de transferencia V_{out} vs. *Caudal* del generador utilizado se muestra en la figura 3.5.

Debido a la saturación de la tensión de salida a partir de 4 L/min no hubiera sido posible establecer una relación directa entre el caudal y el voltaje generado mediante el uso de tablas de equivalencias o *look-up tables*. Además, no es confiable la repetitividad de la función de transferencia para distintas partidas del mismo producto. Es por ello que se decidió el uso de un sensor de rotación. Notar que en la misma figura se indican tres valores típicos de caudal (mínimo, medio y máximo) para dos tipos de grifos en un hogar. Estos valores, comparados con la función de transferencia ponen en evidencia la inviabilidad del uso de *look-up tables*.

Grifo #1 (indicado con estrellas)

- Ubicación en el hogar: frente/fachada
- Uso: riego jardín
- Posición en red de cañería: entre suministro público (taque maestro/barrial) y tanque hogar

Grifo #2 (indicado con cruces)

- Ubicación en el hogar: bacha en sector lavado
- Uso: lavado de ropa, elementos del jardín, limpieza, otros
- Posición en red de cañería: ramal de tanque hogareño

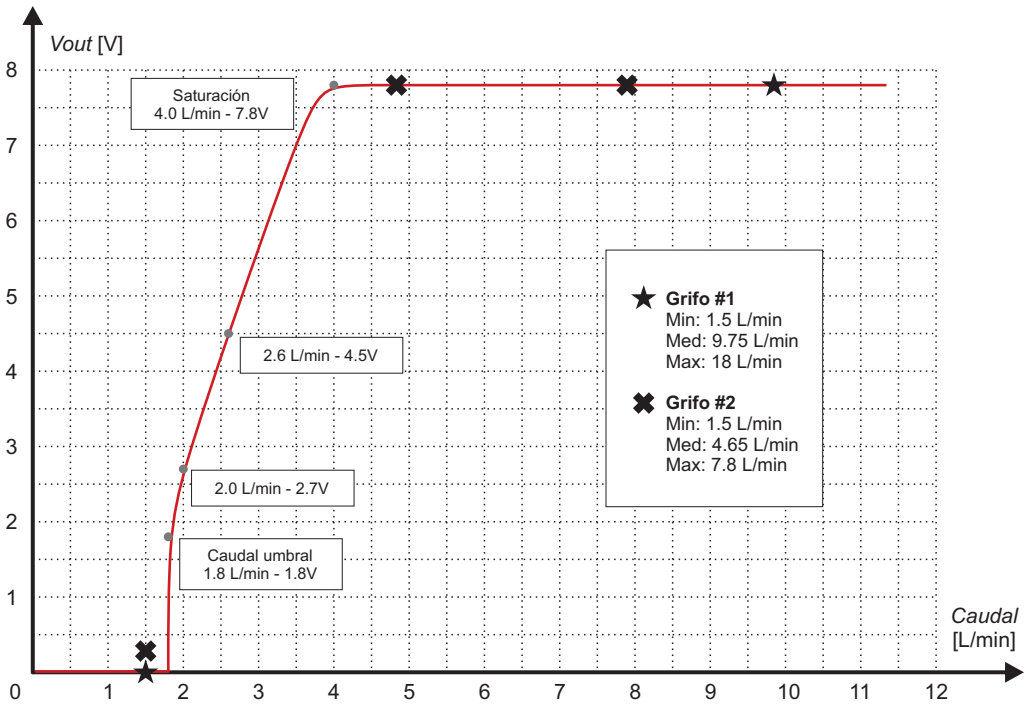


Figura 3.5: Función de transferencia V_{out} vs. *Caudal*.

Notar que el generador comienza a girar con un caudal de 1.8 L/min, lo que significa que caudales menores a éste no podrán detectarse si no se cuenta con algún dispositivo extra de medición de bajo caudal y un sistema de almacenamiento y gestión de energía para poder tomar sus lecturas cuando el generador no esté funcionando. Esta situación, sumada a la necesidad de valores específicos de voltaje para la electrónica, se analiza con más detalle cuando se plantean distintas alternativas en la sección 3.1.3.

3.1.2. Sensor de campo magnético

Medidor.SensorMagnético.*IdeaDesign*

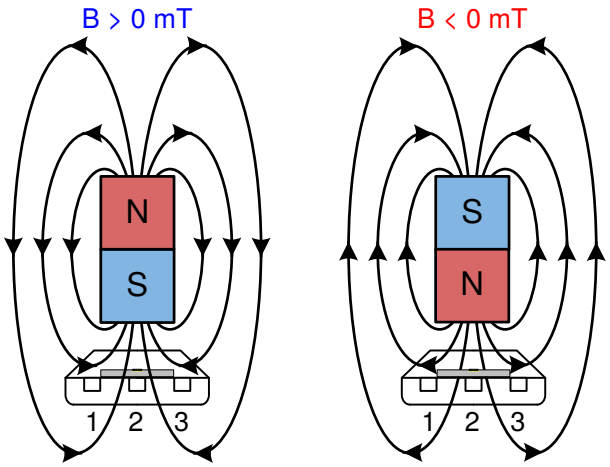
Este elemento es necesario para medir las revoluciones del generador, relacionadas al caudal circulante por el mismo.

Se eligió un sensor de *efecto Hall* para detectar el movimiento del rotor húmedo. Debido a que este rotor tiene contacto directo con el flujo de entrada, se consideró acertado medir sus revoluciones en lugar de hacerlo en el rotor seco, el cual puede sufrir desacoples debido a aumentos en la inercia del motor, provocados por variaciones de consumo por parte de la electrónica conectada al mismo (transitorios del circuito de acondicionamiento, transmisión y otros).

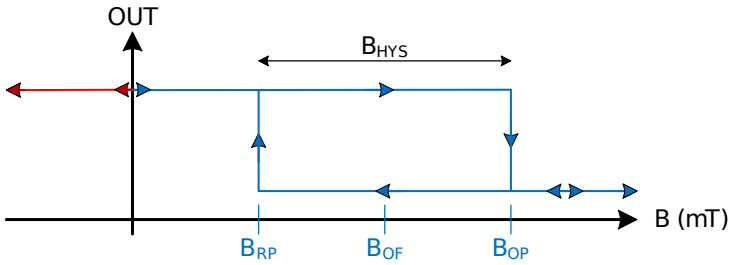
Como se muestra en la figura 3.6b, cuando la densidad de flujo magnético B aplicado excede el valor umbral B_{OP}, la salida en drenador abierto del dispositivo va hacia abajo. La salida permanece baja hasta que el campo disminuye a un valor menor a B_{RP}, y la salida cambia a alta impedancia.

Al ser la salida de tipo drenador abierto es necesario colocar un resistor de valor recomendado entre 1 $k\Omega$ y 10 $k\Omega$ para que cumpla la función de *pull-up*, significando esto conectar la tensión V_{ref} a la salida del dispositivo cuando el transistor de efecto de campo (FET, *Field Effect Transistor*) desconecta el trayecto hacia GND (figura 3.7).

Como se ve en la figura 3.8 se colocó un resistor de *pull-up* $R1$ de 4.7 $k\Omega$ y un capacitor de filtrado $C1$ de 100 nF.



(a) Orientación del campo magnético.



(b) $B(mT)$ vs. Salida.

Figura 3.6: Comportamiento del sensor de efecto Hall.

Características:

- Tipo: *switch* unipolar
- Sensibilidad: $B_{OP} = 6.9\ mT$; $B_{RP} = 3.2\ mT$
- Histéresis: $B_{hys} = 3.7\ mT$
- Rango de voltaje: $2.5 - 38\ V$

Medidor.SensorMagnético.*PartsOnHand*

- Modelo: DRV5023AJQ
- Fabricante: Texas Instruments
- Encapsulado: SOT23

Medidor.SensorMagnético.*Assembling*

En el modelo 3D de la figura 3.10 se ve el sensor insertado en proximidad al rotor húmedo. Se envolvió el sensor en tubo termocontraible para aislarlo del líquido y así evitar cortocircuitos (ver figura 3.11).

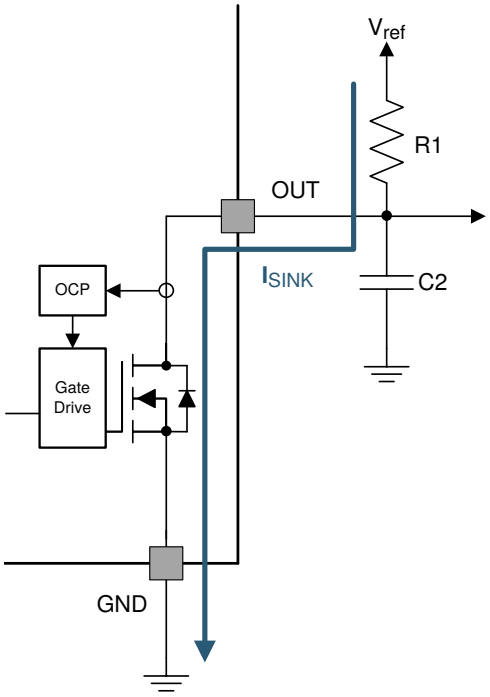


Figura 3.7: Esquemático en *datasheet*.

Medidor.SensorMagnético. *Testing*

Antes de tomar la decisión de utilizar un sensor de efecto Hall se realizaron pruebas del mismo sensando las revoluciones del prototipo motor-generador acoplados por engranajes de la figura 3.12a. Allí mismo se puede ver al sensor (sostenido con la mano) en la proximidad del engranaje mayor, con el fin de capturar el campo magnético de un pequeño imán girando a la velocidad del sistema improvisado.

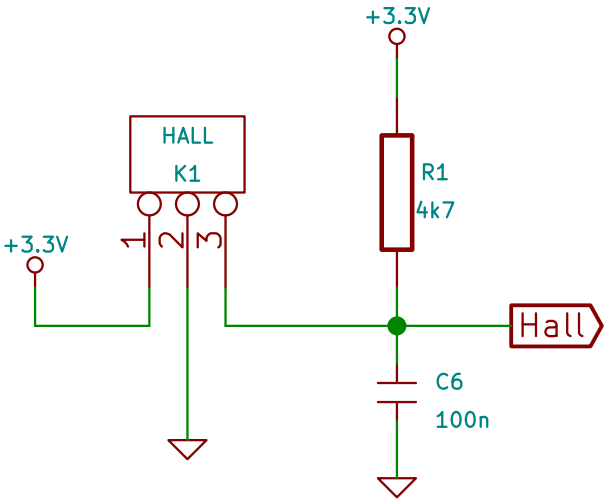


Figura 3.8: Esquemático en *software* KiCad

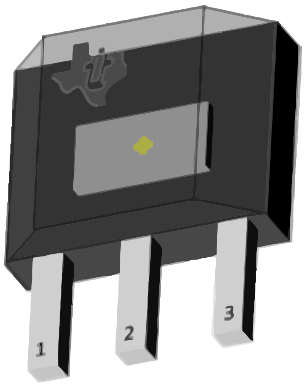
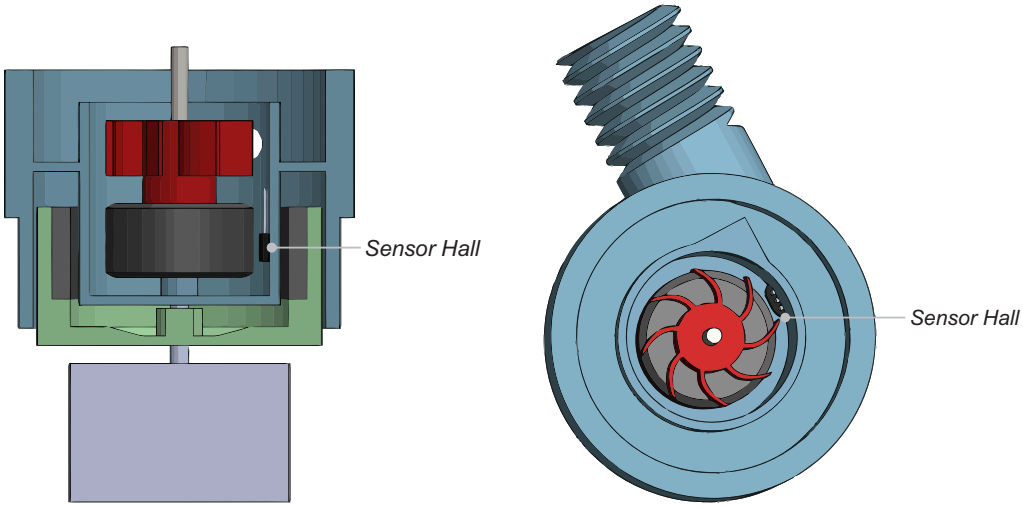


Figura 3.9: Sensor de efecto Hall en encapsulado SOT23.



(a) Corte. (b) Vista superior.

Figura 3.10: Modelo 3D de sensor instalado en generador.

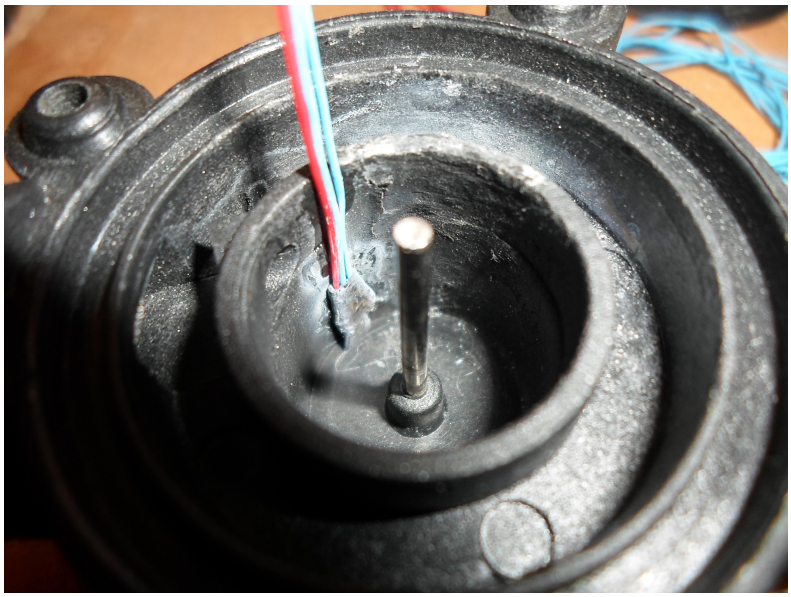
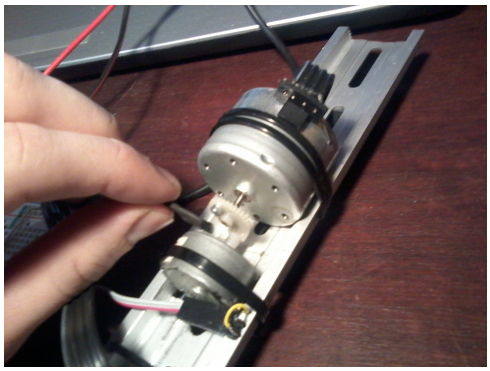
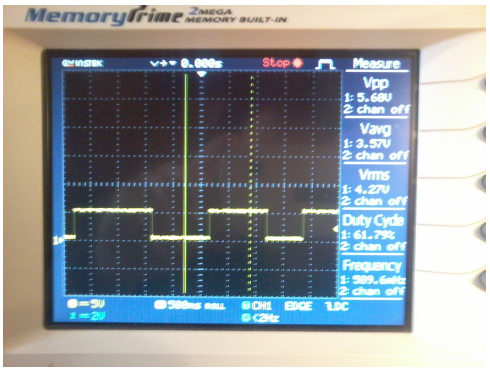


Figura 3.11: Fotografía del sensor Hall instalado en la cámara del rotor húmedo.



(a) Sensor de efecto Hall en prototipo de prueba motor-generador.



(b) Tensión de salida tipo *switch*— VCC o GND.

Figura 3.12: Prueba de sensibilidad y respuesta del sensor de efecto Hall.

3.1.3. Acondicionamiento de potencia

Medidor.Acondicionamiento.*IdeaDesign*

Previo al desarrollo del sistema elegido, se consideraron distintas alternativas para el diseño de este bloque. Para lograr una recolección de energía de manera satisfactoria se tuvieron en cuenta distintos factores como funcionalidad y complejidad del diseño, precios de los componentes electrónicos y disponibilidad nacional e internacional, en ese orden de prioridad.

Se tomó como punto de partida para el análisis del acondicionamiento de potencia la función de transferencia del generador mostrada en la figura 3.5. La curva se redibuja linealizada por tramos y se divide en tres etapas con el fin de simplificar su análisis y explicación. En la primera etapa, correspondiente al tramo *LOW* de la figura 3.13, el caudal no es suficiente para hacer girar al generador, por lo que no hay voltaje generado ni caudal que medir. Este escenario supone uno de los desafíos más importantes de este trabajo. A continuación se proponen tres alternativas en el manejo de la energía para poder cumplir con las soluciones propuestas al inicio de este informe.

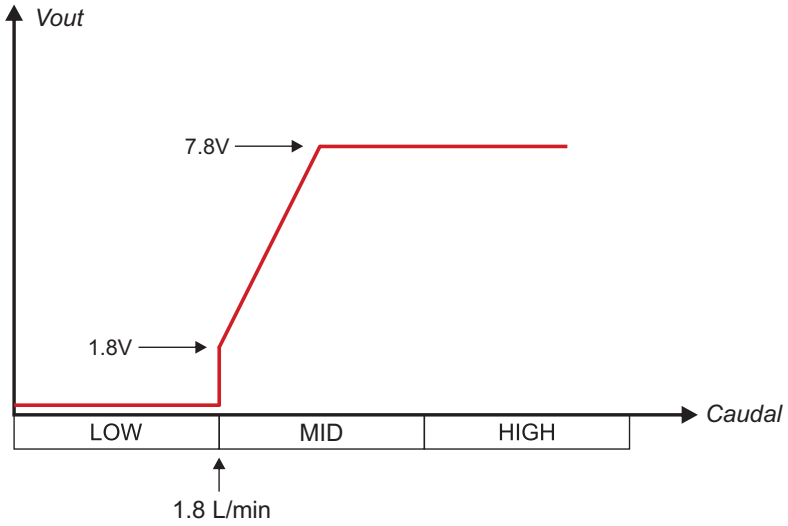


Figura 3.13: Función de transferencia simplificada y dividida en etapas.

La alternativa A (figura 3.14) plantea el uso de un medidor de caudal (figura 3.18 en serie al generador y un sistema de almacenamiento basado en un capacitor tipo *EnerChip* [6]. Esta configuración permite ampliar el rango de medición hacia valores bajos de caudal (menores a 1.8L/min que es el mínimo posible para el generador de la figura 3.13). Para lograrlo, el sistema necesita un *back-up* de energía para que el microcontrolador tome las mediciones cuando el generador no esté funcionando (situación de la figura 3.15). Los capacitores tipo *EnerChip* (figura 3.17) son fabricados con material semiconductor y poseen dos características muy apropiadas para este desarrollo: miles de ciclos de carga-descarga y una corriente de pérdida muy baja. Esta última característica garantiza el mantenimiento de la energía entre sesiones prolongadas. Se entiende por sesión al conjunto de eventos consecutivos que ocurren durante el funcionamiento del sistema: *sin caudal*, *caudal sin generación de energía* (figura 3.15), *caudal con generación de energía* (figura 3.16), *sin caudal*.

Esta alternativa sigue ofreciendo la característica diferencial de no necesitar mantenimiento asociado al recambio de baterías y además cuenta con un gran rango de medición.

La desventaja de esta configuración es el costo asociado a componentes adicionales como el sistema de almacenamiento y el medidor en serie.

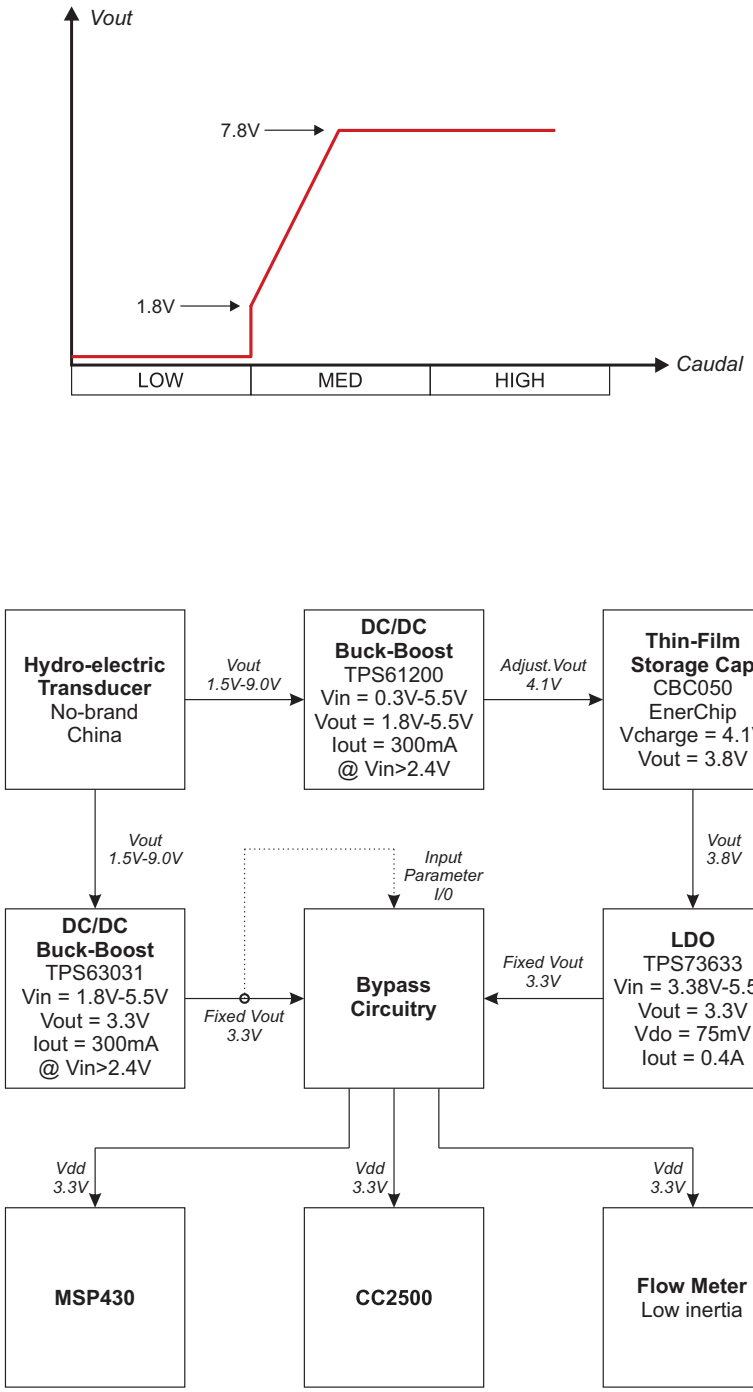


Figura 3.14: Alternativa A: *Buck-Boost* + Capacitor + Medidor en serie.

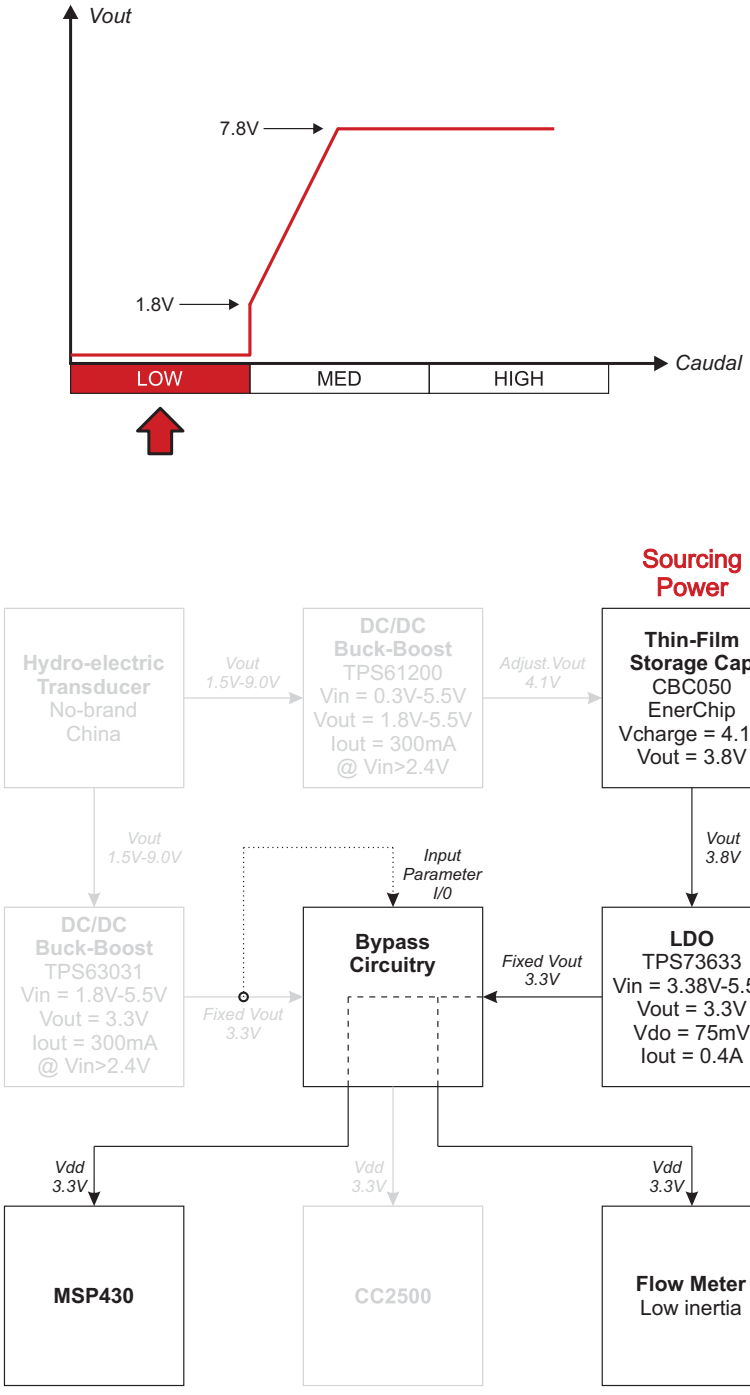


Figura 3.15: Alternativa A. Etapa 1.

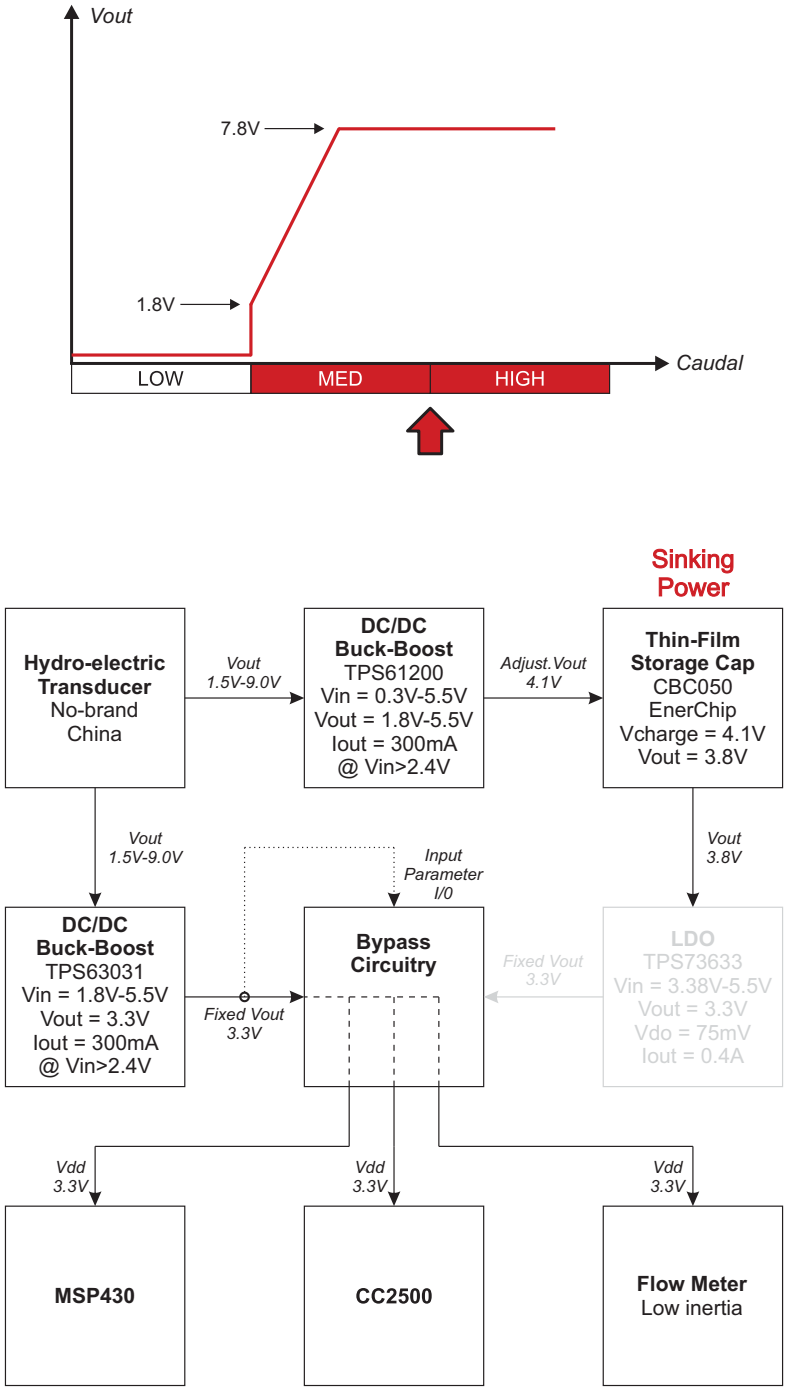


Figura 3.16: Alternativa A. Etapa 2.

Figura 3.17: Capacitores de almacenamiento *EnerChip*.

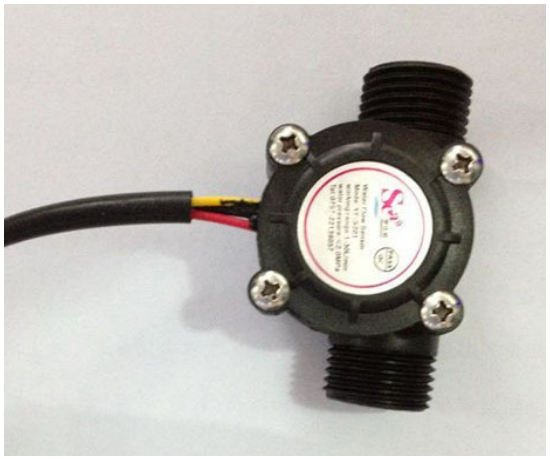
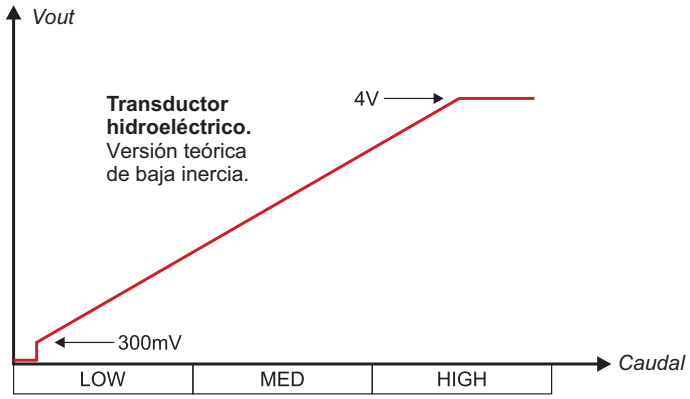


Figura 3.18: Medidor de caudal por pulsos.

La alternativa B (figura 3.19) plantea el uso de un generador teórico de baja inercia que permita generar voltaje (y por ende detectar rotación debido al giro) a caudales menores a 1.8 L/min para poder cubrir totalmente el rango de caudales típicos en un hogar. Esta alternativa supone la existencia de un generador que no está disponible en el mercado. Sin embargo, científicos del departamento de ingeniería mecánica en la Universidad Kun Shan en Taiwan han diseñado y desarrollado un generador prototipo (figura 3.20) que genera energía a muy bajo caudal [7]. El mismo es de una fabricación sencilla. A pesar de ello, sería necesario el uso de una impresora 3D para fabricar el transductor que materializaría la alternativa B.

Como se ve en la figura 3.21, existe un caudal no detectable muy bajo en comparación al caudal mínimo del generador de la alternativa A. Durante la segunda etapa (figura 3.22), mediante el uso de un convertidor DC-DC *buck-boost*, se mantiene constante una tensión de salida de 3.3 V en un rango de V_{in} entre 300 mV y 5.5 V



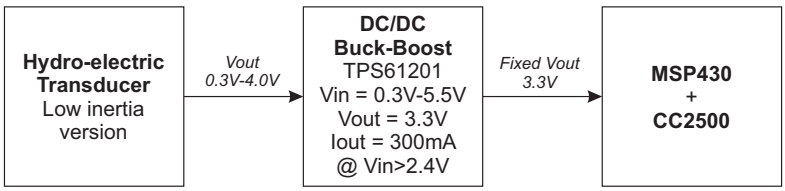
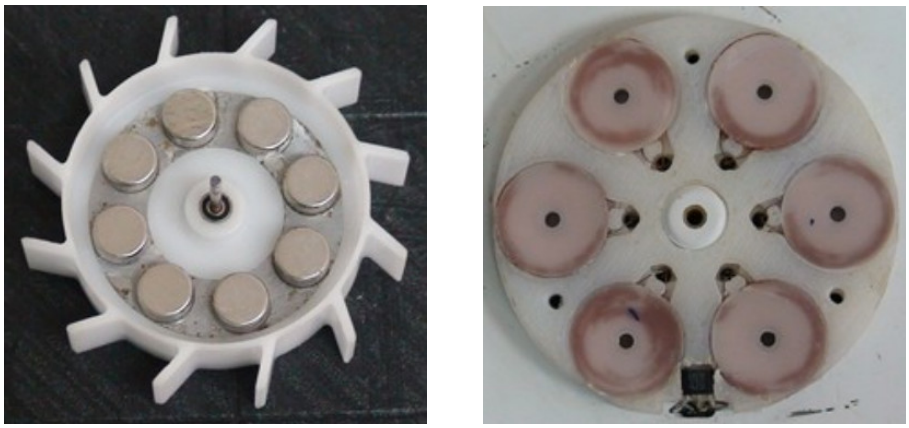


Figura 3.19: Alternativa B: Generador teórico de baja inercia + *Buck-Boost*.



(a) Rotor. (b) Estator.

Figura 3.20: Prototipo de generador AFPM (*Axial Flux Permanent Magnet.*)

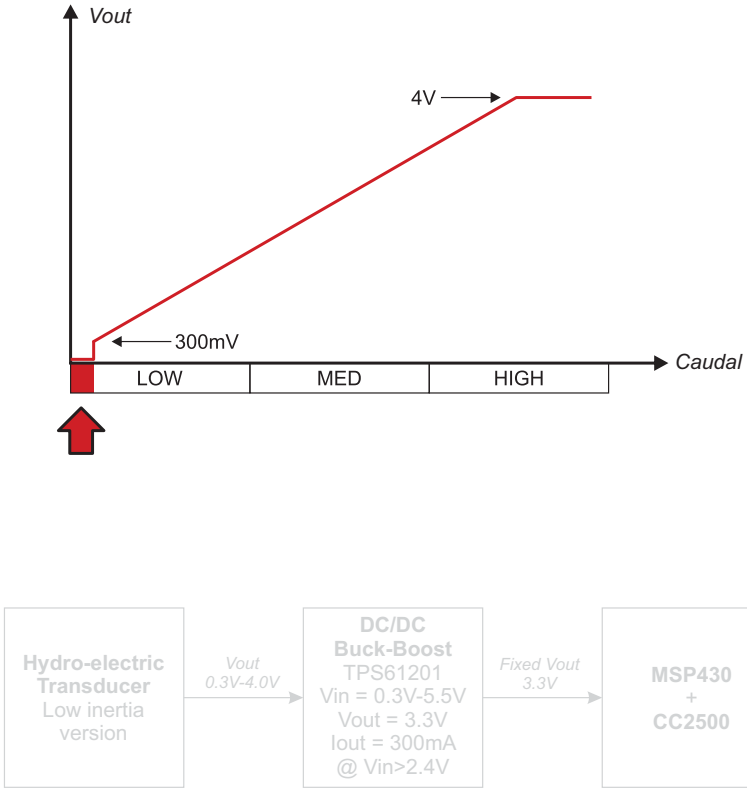
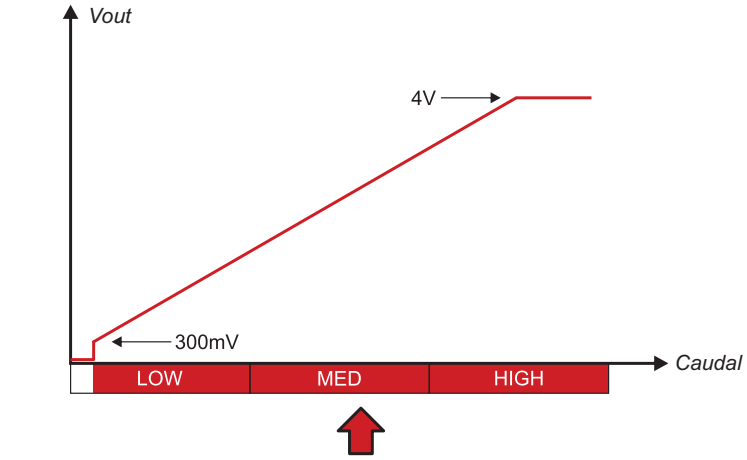


Figura 3.21: Alternativa B. Etapa 1.



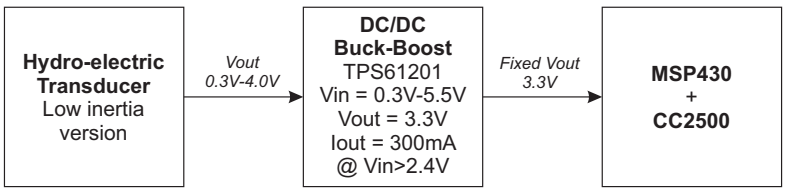
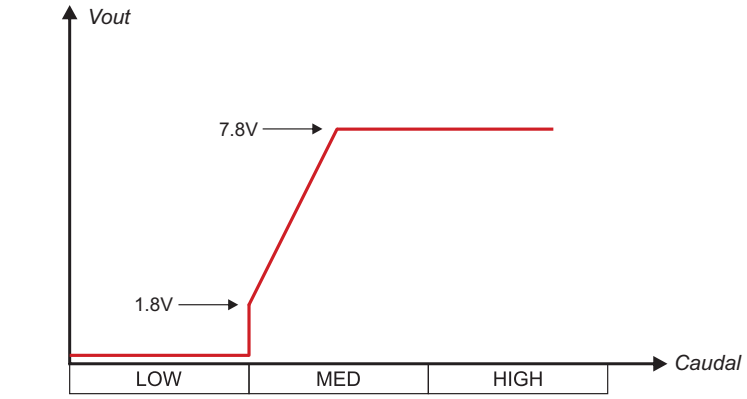


Figura 3.22: Alternativa B. Etapa 2.

La alternativa C (figura 3.23) ofrece una solución sencilla, robusta y de bajo costo en comparación a las anteriores, sacrificando la medición de una porción de caudal (entre 1.8 L/min y 2.2 L/min) lo cual es insignificante frente a los caudales típicos usados en un hogar como se vio en la figura 3.5.

En la primera etapa (figura 3.24), el generador no gira hasta que llega al caudal umbral de 1.8 L/min. De allí hasta aproximadamente 2.2 L/min el generador gira pero su energía no es suficiente para que la electrónica funcione de manera correcta, por lo que el sistema es habilitado recién cuando se generan 3.38 V, voltaje mínimo necesario para que el regulador pueda ofrecer 3.3 V (figura 3.25).



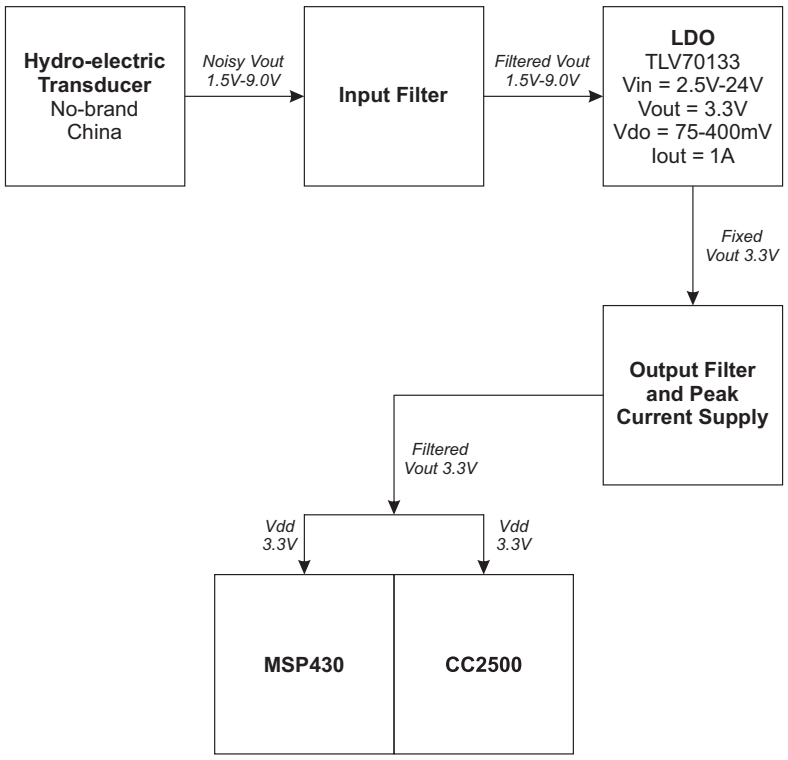


Figura 3.23: Alternativa C: Regulador lineal de bajo *drop-out* + Filtro.

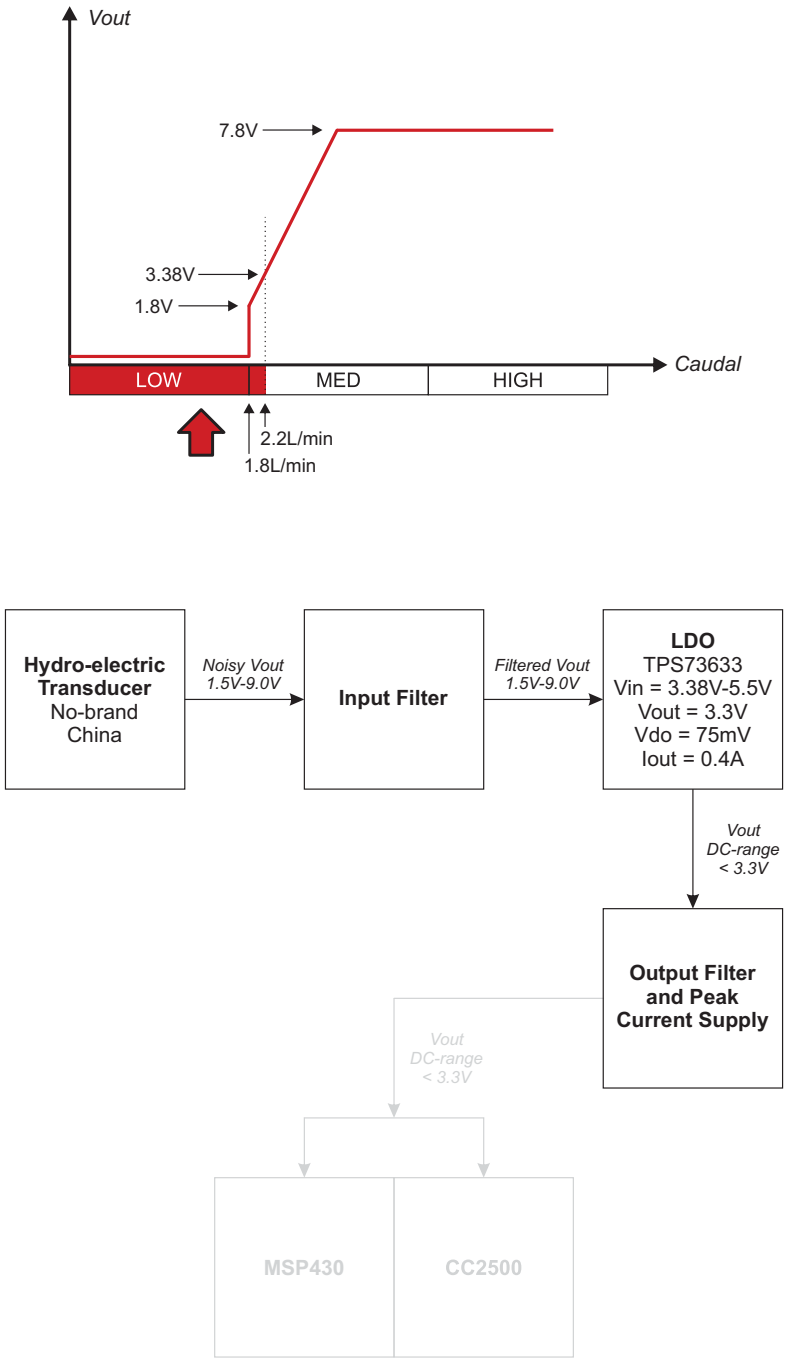


Figura 3.24: Alternativa C. Etapa 1.

Gracias a la existencia de reguladores lineales de baja tensión de *drop-out*, esto es, el voltaje mínimo necesario por sobre el voltaje de regulación, es viable su uso como alternativa en este tipo de escenario, reemplazando el uso de un convertidor DC-DC tipo *switching*, lo cual trae asociado complejidad y costo en el diseño.

Debido a que el rango de caudal omitido por esta alternativa (desde 1.5 L/min hasta 2.2 L/min) es relativamente estrecho (4.2 %) en relación al rango completo de caudales típicos en un hogar (desde 1.5 L/min hasta 18 L/min) se seleccionó la alternativa C como apropiada para desarrollar. Esta alternativa es capaz de cumplir las especificaciones del bloque de acondicionamiento de potencia.

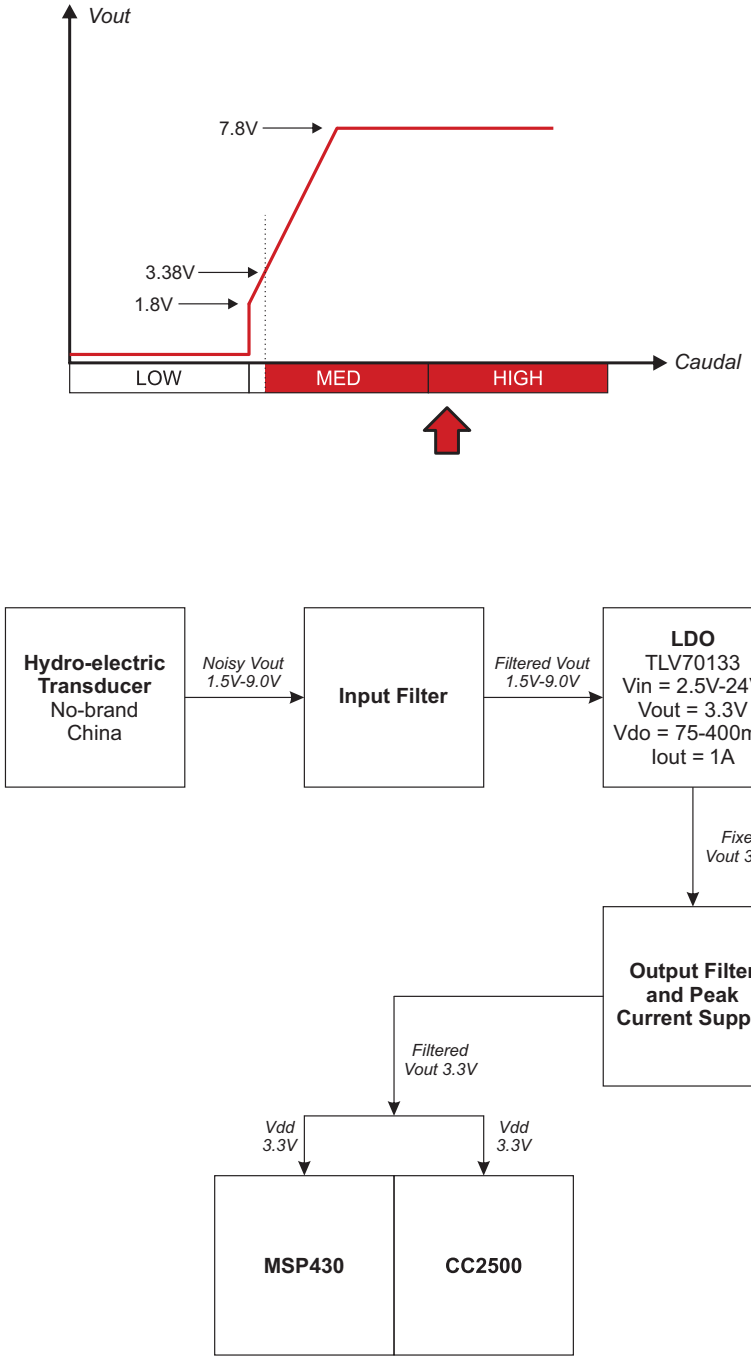


Figura 3.25: Alternativa C. Etapa 2.

Medidor.Acondicionamiento.*PartsOnHand*

Regulador lineal de bajo drop-out:

- Modelo: TLV70133
- Fabricante: Texas Instruments
- Encapsulado: SOT23 (5-lead)
- Precio: USD 0.96

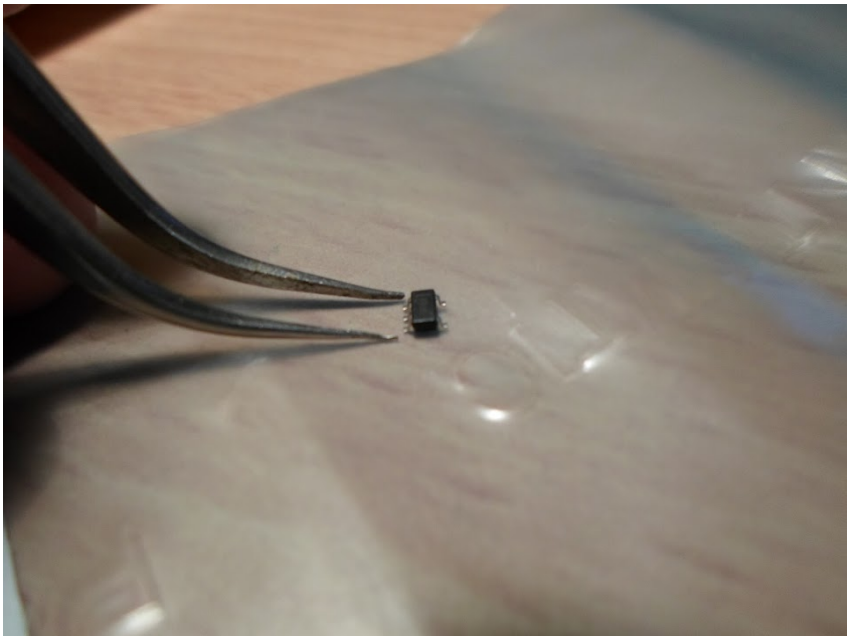


Figura 3.26: TLV70133 al lado de pinza para SMD.

Medidor.Acondicionamiento.*Assembling*

A la hora de diseñar el PCB (figura 3.27) se tuvo en cuenta el espacio que ofrecía la carcasa del generador (ver figura 3.27b) con el fin de utilizar esta misma para albergar el resto de la electrónica, logrando así una disminución de costos asociado a un gabinete extra y una estética mucho más profesional.

Los componentes R_1 y C_6 correspondientes al bloque *Sensor de Campo Magnético* (figura 3.8) se colocaron en este circuito impreso.

Medidor.Acondicionamiento.*Testing*

El fabricante del regulador lineal recomienda utilizar valores de capacidad de $1\mu F$ o más tanto para la entrada como para la salida del mismo. Sin embargo, debido a que el voltaje entregado por el generador es bastante ruidoso, los valores de capacidad (sobretodo en la entrada) fueron elegidos de manera experimental hasta obtener una tensión de salida aceptable. En la captura de osciloscopio de la figura 3.28 puede verse en el canal uno (curva superior) el voltaje generado sin regular y sin filtrar, mientras que en el canal dos (curva inferior) puede verse el voltaje regulado y filtrado.

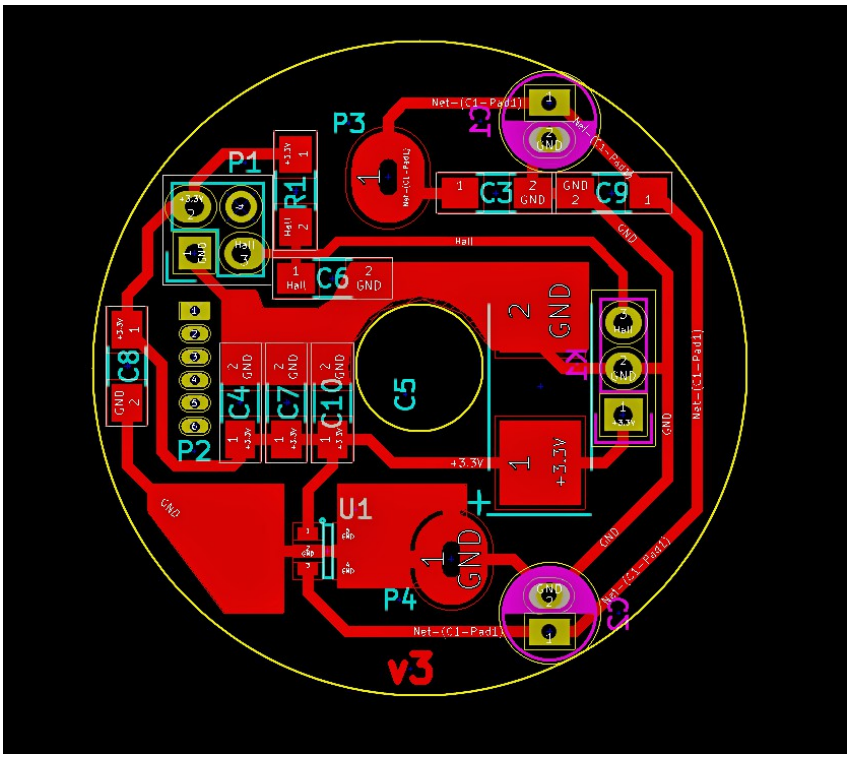
El diseño final basado en recomendaciones del fabricante y experimentos en el laboratorio se muestra en la figura 3.29. El valor total de capacidad corresponde a la suma de los valores individuales de capacidad dispuestos en paralelo. Para el lado de la entrada (antes del dispositivo regulador) la capacidad total es:

$$\begin{aligned} C_{T_{in}} &= C_1 + C_2 + C_3 + C_9 \\ C_{T_{in}} &= 100\mu F + 100\mu F + 100\eta F + 100\eta F \\ C_{T_{in}} &= 200.2\mu F \end{aligned} \tag{3.1}$$

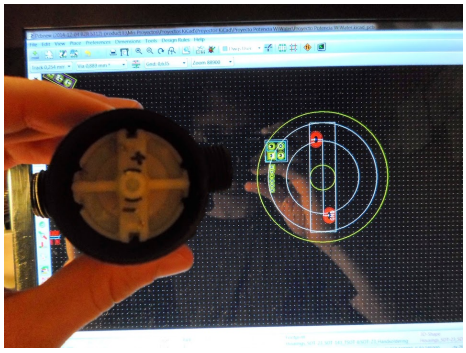
Lo mismo sucede para el filtro de salida (después del dispositivo regulador) que se compone de los capacitores C_4, C_5, C_7, C_8 y C_{10}.

$$
\begin{aligned}
C_{T_{out}} &= C_4 + C_5 + C_7 + C_8 + C_{10} \\
C_{T_{out}} &= 100\eta F + 47\mu F + 100\eta F + 100\eta F + 100\eta F \\
C_{T_{out}} &= 47.4\mu F
\end{aligned} \tag{3.2}
$$

En los terminales P_3 y P_4 que se muestran en el esquemático de la figura 3.29 se conectan los bornes de salida del generador. Antes de llegar a la versión definitiva de la figura 3.27d se fabricaron otras dos versiones. Luego de encontrar un valor suficiente de capacidad, tanto a la entrada como a la salida, el siguiente desafío consistió en ubicar los capacitores electrolíticos C_1 y C_3 de tal forma que permitan el ensamble del resto de la electrónica.

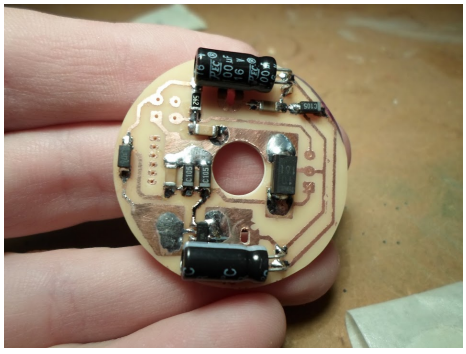


(a) *Layout*.

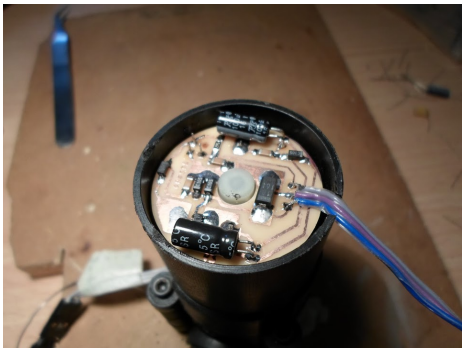


(b) Cálculo de espacio.

(c) Montaje de prueba.



(d) PCB con componentes.



(e) Placa soldada al generador.

Figura 3.27: Diseño y fabricación del *PCB* de acondicionamiento de potencia.

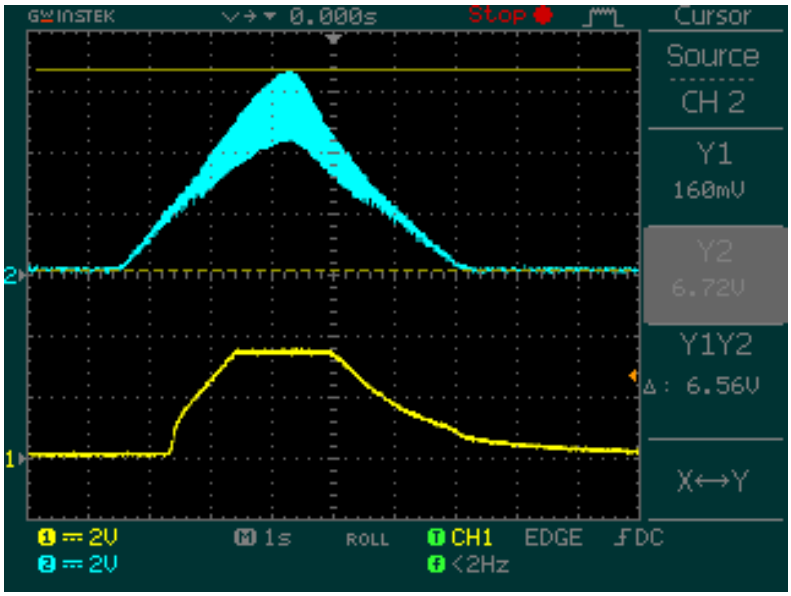


Figura 3.28: Arriba: voltaje sin filtrar y sin regular. Abajo: voltaje filtrado y regulado.

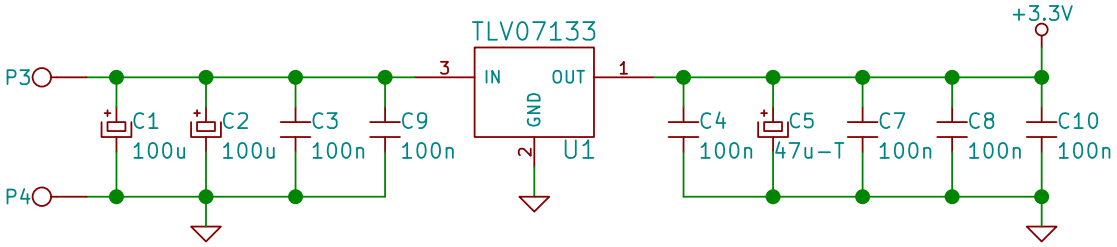


Figura 3.29: Regulador lineal de bajo *drop-out* y capacitores de filtrado.

3.1.4. MCU—microcontrolador

Medidor.MCU.*IdeaDesign*

Los MSP430F2x/4x, pertenecientes a la serie *MSP Ultra-Low-Power MCU* de la firma Texas Instruments, son microcontroladores de propósito general de 16-*bits* usados en un amplio rango de aplicaciones incluyendo electrónica de consumo, *data logging*, instrumentos médicos portables, y medición de bajo consumo.

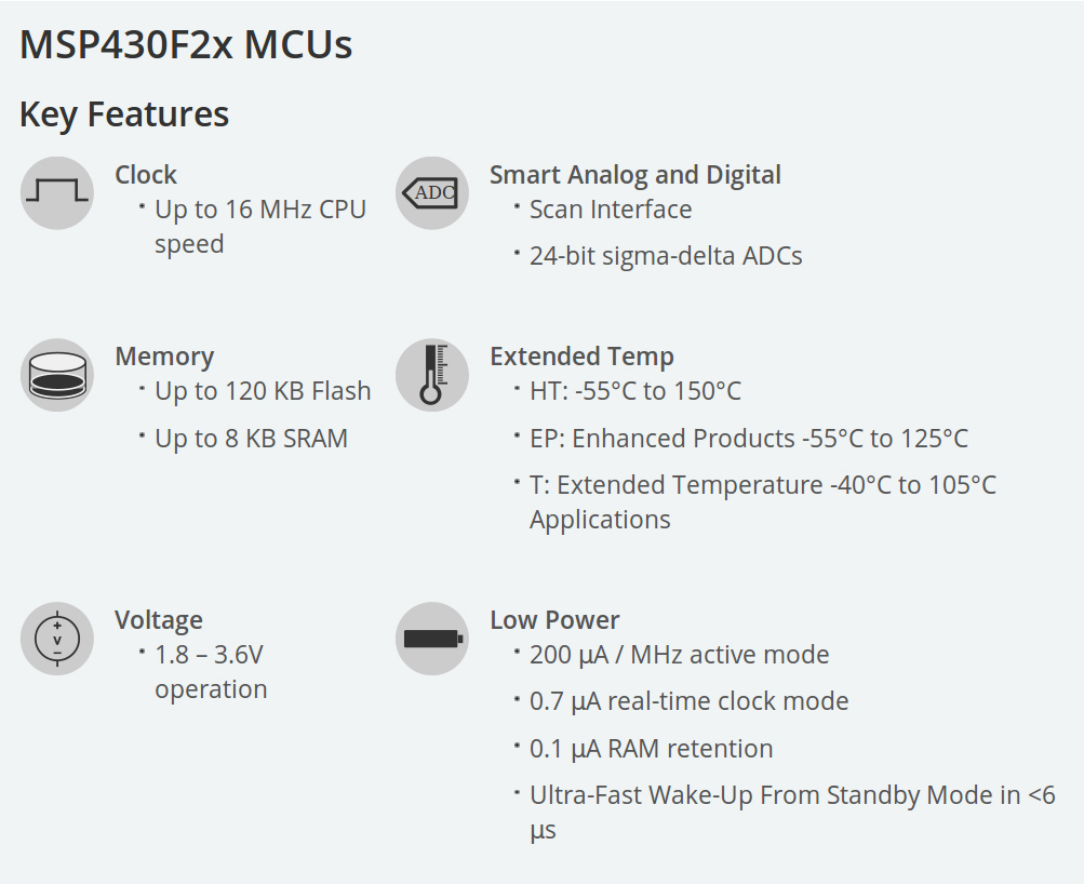


Figura 3.30: *MSP430F2x MCUs Key Features* [8].

Las características principales de la serie MSP430F2x (figura 3.30) se ajustan a los requerimientos más importantes del sistema como son memoria flash para almacenar mediciones relativas entre sesiones, procesamiento de bajo consumo, módulos de comunicación, entre otros.

En el diagrama de la figura 3.31 se muestran los bloques funcionales que componen la estructura del microcontrolador elegido: el MSP430F2274.

Medidor.MCU.*PartsOnHand*

Se seleccionó específicamente una placa de desarrollo de Texas Instruments que ofrece dos soluciones en un mismo circuito impreso: microcontrolador y transceptor de radiofrecuencia con antena integrada, por lo que el *hardware* de los bloques **MCU**, ***Transceiver*** y **LED para *debugging*** está resuelto en el dispositivo eZ430-RF2500T de la figura 3.32. En el esquemático de la figura 3.33 se puede ver la interconexión entre el microcontrolador, el *transceiver*, el circuito de adaptación de antena, la antena integrada y el resto de

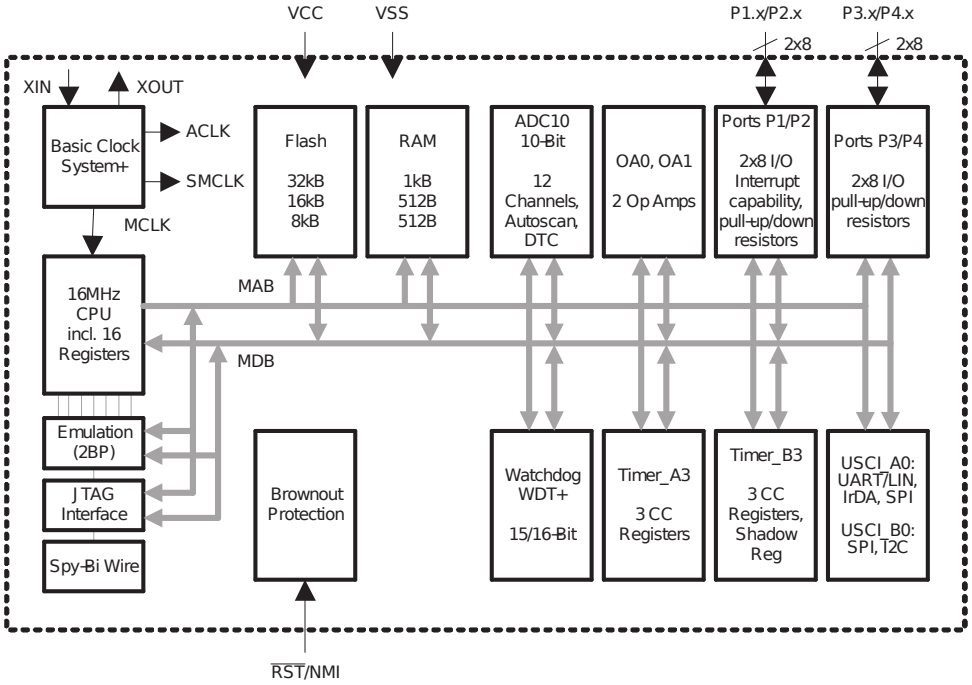


Figura 3.31: Bloques funcionales del microcontrolador MSP430F2274 [9].

los componentes de la placa eZ430-RF2500T.

La lectura de los pulsos provenientes del sensor de efecto Hall se toman en el pin P2.0 del MSP430 en la zona de expansión como se ve en el esquemático de la figura 3.33.

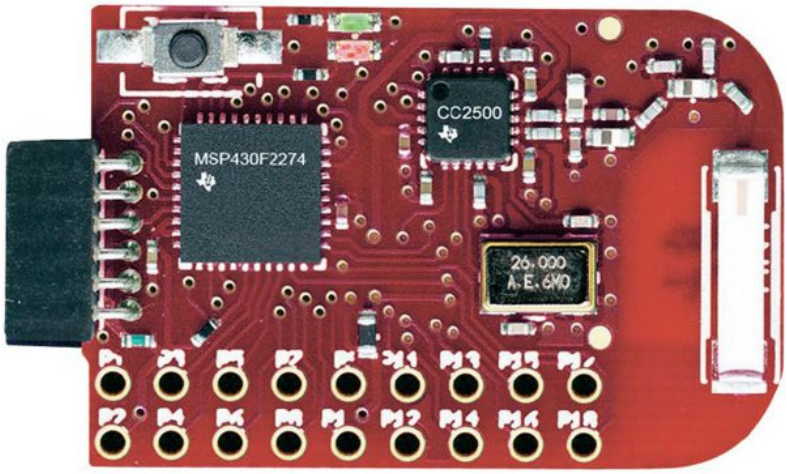


Figura 3.32: eZ430-RF2500T: MSP430F2274 + CC2500.

Medidor.MCU.*Assembling*

La placa eZ430-RF2500T posee un microcontrolador MSP430F2274 y un *transeiver* CC2500 con adaptación de antena y antena-chip para 2.4 GHz, pulsador y LEDs para *debugging* y pines de expansión y grabación. Esta herramienta de desarrollo posee las características justas para la solución planteada tanto por sus prestaciones como por sus dimensiones ya que se puede alojar en el gabinete del generador junto a la placa de acondicionamiento presentada en la sección 3.1.3 (figuras 3.34).

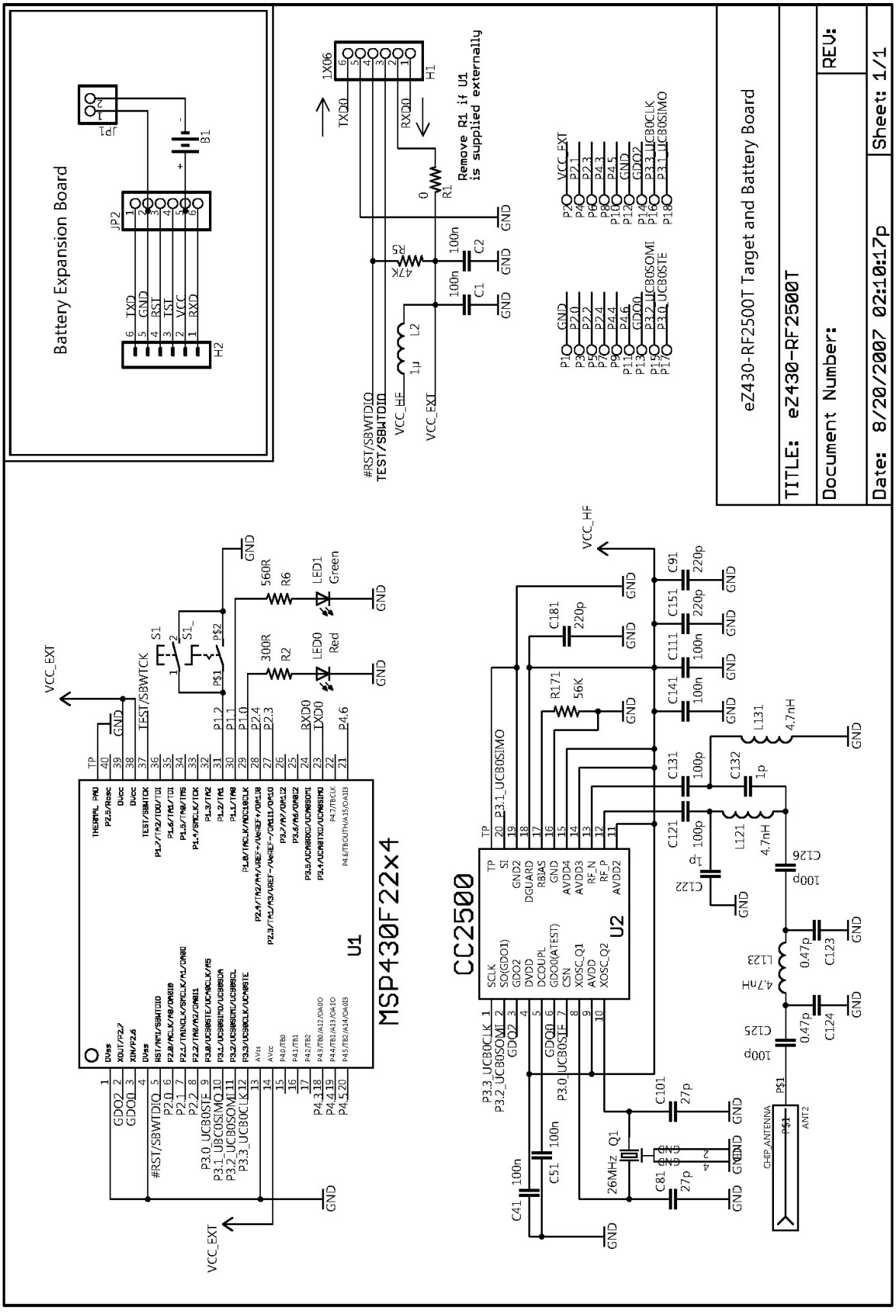
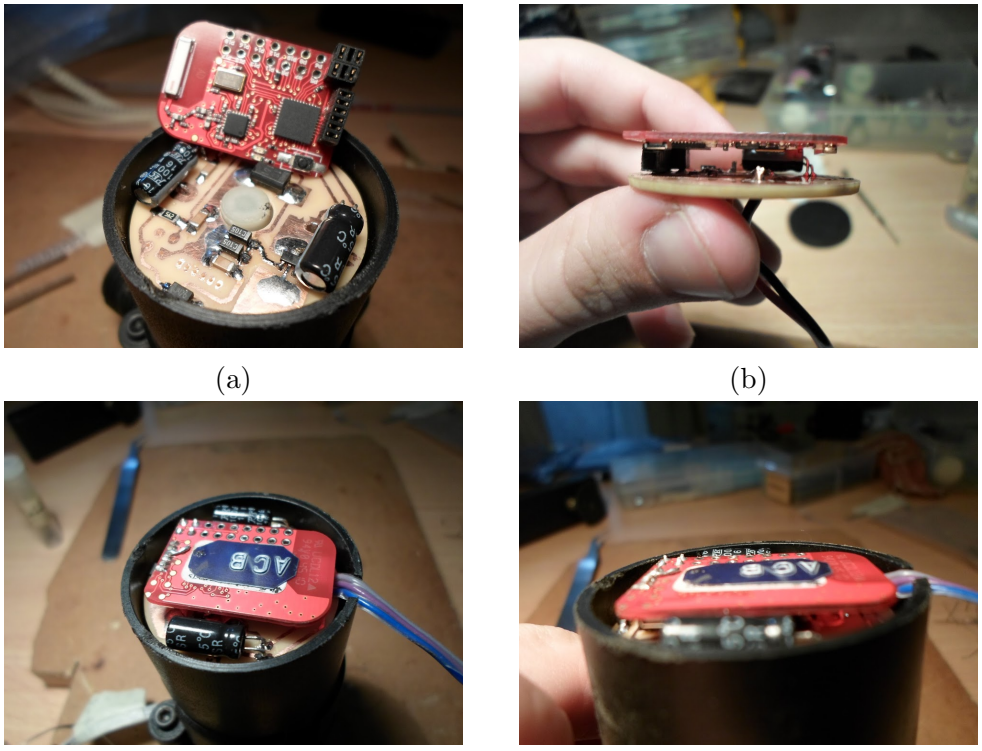


Figura 3.33: Esquemático de conexión entre MSP430F2274 y CC2500 [10].



(a) (b)

(c) (d)

Figura 3.34: eZ430-RF2500T ensamblado a PCB de acondicionamiento.

Se agregó un conector de cuatro contactos a la hilera de pines de expansión del eZ430-RF2500T (figura 3.34a) con el objeto de acoplar mecánica y eléctricamente las dos placas: el PCB de acondicionamiento y el eZ430-RF2500T. La placa de acondicionamiento le suministra al resto de la electrónica alimentación regulada y filtrada y los pulsos provenientes del sensor de efecto Hall para que el microcontrolador pueda procesarlos.

Medidor.MCU. *Testing*

Para probar el buen funcionamiento del ensamblaje de la figura 3.32 se conectó el generador a la red de agua y mediante un *firmware* de prueba se hizo parpadear el LED de *debugging* para indicar una correcta alimentación y recepción de los pulsos del sensor de efecto Hall. En la captura de osciloscopio mostrada en la figura 3.35 se puede apreciar en la curva superior la salida tipo *on/off* del sensor de campo magnético, mientras que en la curva inferior se ve la alimentación filtrada y regulada.

Notar que la señal proveniente del sensor Hall *copia* la señal de alimentación ya que el dispositivo es alimentado por la misma tensión que todo el sistema. Analizando en detalle la curva superior se puede notar que a mayor tensión generada, mayor es la velocidad de rotación. Esto se puede comprobar al ver que la densidad de pulsos es mayor en el centro de la curva. También es notable cómo el dispositivo sensor comienza a funcionar a partir de 2 V aproximadamente.

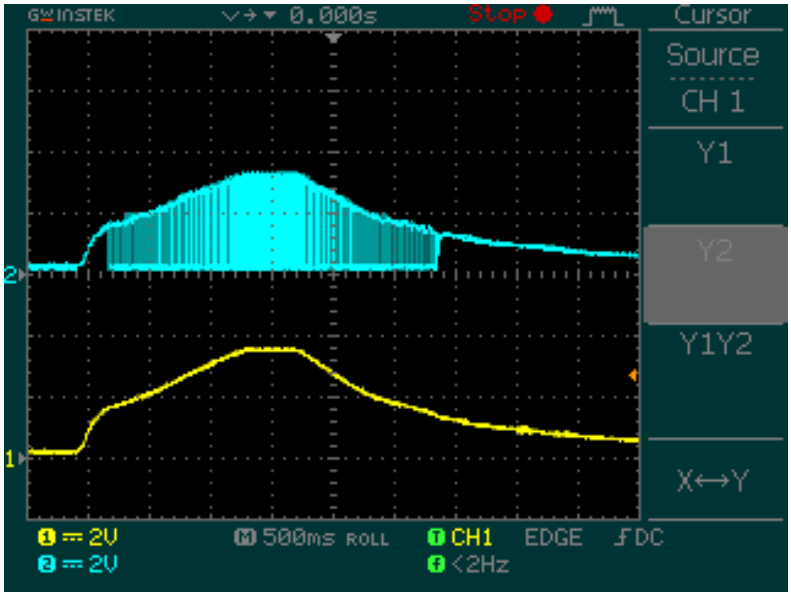


Figura 3.35: Arriba: salida del sensor Hall. Abajo: alimentación del sistema.

3.1.5. *Transceiver* y circuito de adaptación de antena

Medidor.*Transceiver.IdeaDesign*

El CC2500 es un *transceiver* de 2.4 GHz de bajo costo diseñado para aplicaciones inalámbricas de muy bajo consumo. El dispositivo está pensado para trabajar en la banda de frecuencia 2400-2484.5 MHz ISM (Industrial, Científico y Médico) y SRD (Dispositivos de Rango Corto).

El transceptor RF integra un módem altamente configurable. El módem soporta varias formas de modulación y posee una velocidad de datos configurable de hasta 500 $kBaud$.

El CC2500 provee un extensivo soporte de *hardware* para el manejo de paquetes, *buffering* de datos, transmisión en ráfagas, asignación de canales vacíos, indicador de calidad del enlace y *wake-on-radio* (sistema para despertar el radio).

Los parámetros principales de operación y los *buffers* tipo FIFO (*First Input First Output*) de 64-*byte* de transmisión y recepción pueden ser controlados vía interfaz SPI (*Serial Protocol Interface*).

Características principales:

- Alta sensibilidad (-104 dBm at 2.4 $kBaud$, factor de error en paquetes 1 %)
- Bajo consumo de corriente (13.3 mA in recepción @ 250 $kBaud$)
- Potencia de salida programable hasta +1 dBm
- Excelente *performance* en selectividad y bloqueo
- Velocidad de datos programable de 1.2 a 500 $kBaud$
- Rango de frecuencia: 2400-2483.5 MHz
- 400 nA de consumo de corriente en modo SLEEP
- Rápido tiempo de *startup*: 240 us desde modo SLEEP a modo RX o TX
- Funcionalidad *Wake-on-radio* para *low-power RX polling*
- *Buffers* FIFO de 64-*byte* independientes para RX y TX (permite modo *burst* para transmisión de datos)

Un diagrama de bloques simplificado del CC2500 aparece en la figura 3.36. La señal recibida es amplificada por un amplificador de bajo ruido y luego convertida en cuadratura (I y Q) de manera descendente (*down-converted*) a una frecuencia intermedia (IF, *Intermediate Frequency*). A IF, las señales I/Q son digitalizadas por conversores analógico-digitales (ADC, *Analog to Digital Converter*). El control automático de ganancia (AGC, *Automatic Gain Control*), el filtrado del canal, la demodulación, y la sincronización bit/paquete son realizados de manera digital.

Un cristal es conectado a $XOSC_{Q1}$ y $XOSC_{Q2}$. El oscilador a cristal genera una frecuencia de referencia para el sintetizador, como así también un *clock* para el ADC y la parte digital.

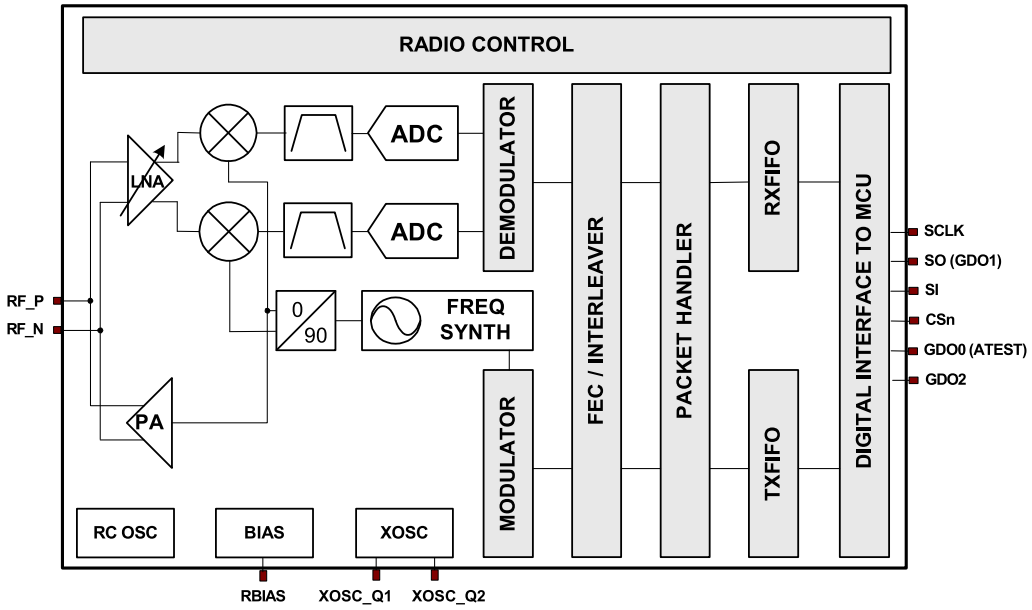


Figura 3.36: Diagrama de bloques del *transceiver* CC2500 [11].

Medidor. *Transceiver.PartsOnHand*

En la tabla de la figura 3.38 se listan los componentes necesarios y sus valores para adaptar la impedancia de la antena-chip (50 Ω) con la entrada del amplificador de bajo ruido y la salida del amplificador de potencia.

Medidor. *Transceiver.Assembling*

Esta sección está unificada a la sección 3.1.4 debido a que el microcontrolador y el *transceiver* se ubican en el mismo circuito impreso. El esquemático del circuito correspondiente al *transceiver* y la adaptación de antena se muestra en la figura 3.37.

Medidor. *Transceiver.Testing*

Se probó el funcionamiento del bloque mediante el uso de un *firmware* ejemplo proporcionado por Texas Instruments. El mismo hace uso de los LEDs de *debugging* y el pulsador de la placa eZ430-RF2500. El ejemplo consiste en una implementación sencilla del protocolo *SimpliciTI* para establecer una comunicación de tipo consulta-respuesta desde un módulo *acces-point* a otro *end-device*. La consulta se ejecuta al presionar el pulsador y mediante los LEDs se observa el resultado de la conversación. Sobre la base de este ejemplo se desarrolla el resto de la aplicación. El protocolo *SimpliciTI* se analiza en la sección 3.2.2.

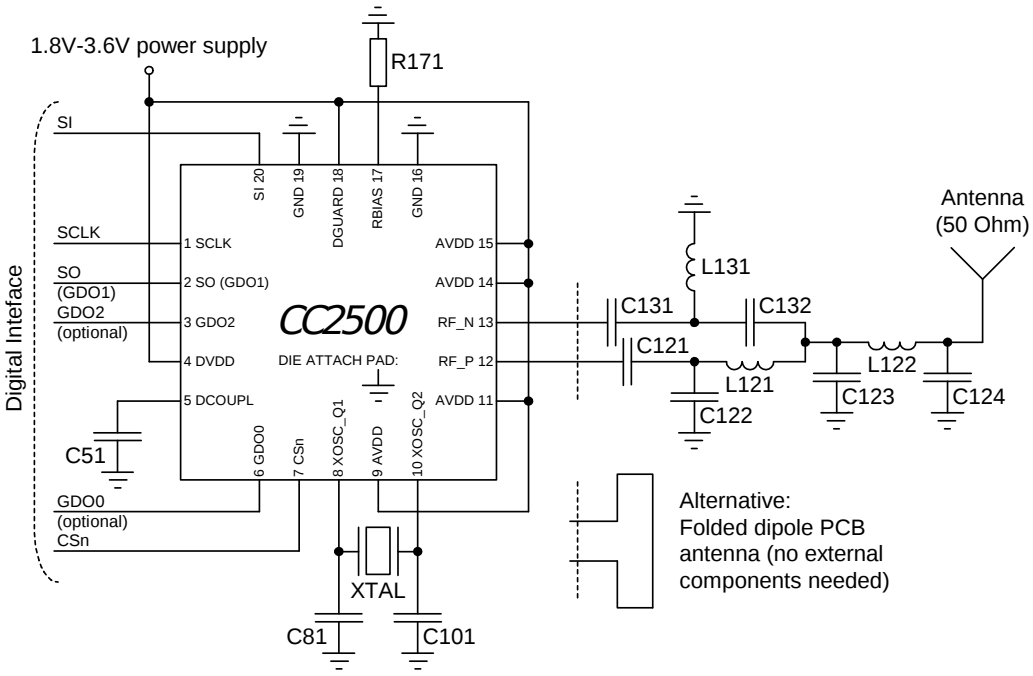


Figura 3.37: Esquemático CC2500 + Adaptación + Antena [11].

Component	Value	Manufacturer
C51	100 nF ±10%, 0402 X5R	Murata GRM15 series
C81	27 pF ±5%, 0402 NP0	Murata GRM15 series
C101	27 pF ±5%, 0402 NP0	Murata GRM15 series
C121	100 pF ±5%, 0402 NP0	Murata GRM15 series
C122	1.0 pF ±0.25 pF, 0402 NP0	Murata GRM15 series
C123	1.8 pF ±0.25 pF, 0402 NP0	Murata GRM15 series
C124	1.5 pF ±0.25 pF, 0402 NP0	Murata GRM15 series
C131	100 pF ±5%, 0402 NP0	Murata GRM15 series
C132	1.0 pF ±0.25 pF, 0402 NP0	Murata GRM15 series
L121	1.2 nH ±0.3 nH, 0402 monolithic	Murata LQG15HS series
L122	1.2 nH ±0.3 nH, 0402 monolithic	Murata LQG15HS series
L131	1.2 nH ±0.3 nH, 0402 monolithic	Murata LQG15HS series
R171	56 kΩ ±1%, 0402	Koa RK73 series
XTAL	26.0 MHz surface mount crystal	NDK, AT-41CD2

Figura 3.38: BOM del CC2500 con adaptación de antena.

3.1.6. LED para *debugging*

Medidor.LED.*IdeaDesign*

Cuando se desarrolla *firmware* o *hardware* es muy valioso contar con elementos de interfaz como LEDs, palabras o símbolos embebidos en el mismo desarrollo, ya que permite *debuggear* con mayor facilidad, esto es, encontrar errores o problemas a medida que construimos el producto final. Estos elementos de interfaz pueden o no cumplir un papel diferente una vez que se ha finalizado el diseño. Para este caso, los LEDs de *debugging* no forman parte de la interfaz de usuario del sistema/producto.

Medidor.LED.*PartsOnHand*

Los elementos de *debugging* (LEDs y pulsador) se encuentran integrados en el circuito impreso del eZ430-RF2500T.

3.2. *Firmware*

En esta sección se presenta el *firmware* del medidor de consumo. El *firmware* o programa se graba y ejecuta en el microcontrolador presentado en la sección 3.1.4. El funcionamiento de dicho programa se describe en el diagrama de flujo de la figura 3.39.

3.2.1. Inicialización de BSP

En este paso se vinculan los elementos de *hardware* (pulsador, LEDs y *transceiver*) a variables y funciones manejables dentro del programa principal. Como se ve en la figura 3.40 el programa principal (ubicado en el archivo `main.c`), accede a los elementos del *hardware* a través de la capa BSP (*Board Support Package*) mediante el manejo de funciones de fácil interpretación e independientes de la naturaleza física y el comportamiento de los dispositivos integrados, como por ejemplo `LED-debugging(ON)`, `Enviar-msg(*trama)`, etcétera. La vinculación entre estos elementos genéricos y el mundo físico correspondiente al *hardware* en el circuito impreso se efectúa en la capa HAL (*Hardware Abstraction Layer*). En resumen, para ordenar y simplificar la programación, el programa principal sólo puede acceder a lo que el archivo `bsp.h` haga visible y a lo que esté por encima de él siguiendo el diagrama de capas de la figura 3.40. La capa BSP se inicializa mediante la función `BSP_Init()`.

3.2.2. El protocolo de la red inalámbrica: SimpliciTI

SimpliciTI es un protocolo de comunicación para redes de sensores inalámbricos diseñado por Texas Instruments, usado en este caso para comunicar los sensores de consumo con el concentrador de datos. Este protocolo se encarga desde configurar la capa física de los transceptores hasta gestionar los paquetes y permisos entre los distintos actores dentro de la red.

La API

La interfaz de programación de aplicaciones (API, *Application Programming Interface*) de SimpliciTI consiste en un conjunto de funciones que permite inicializar, configurar y manejar datos dentro de la red. A continuación se nombra y describe brevemente cada una de estas funciones.

SMPL_Init(): Esta función inicializa el radio y el *stack* del protocolo. Debe ser llamada una vez iniciado el *software* y antes de llamar cualquier otra función de la API.

SMPL_Link(): Esta función envía un *link* en *broadcast* y espera respuesta. Una vez recibida la respuesta una conexión es establecida entre los dos actores y un *link-ID* es asignado al vínculo para ser usado por la aplicación.

Si bien esta función espera por una respuesta, de no llegar la misma en un período de tiempo preestablecido retornará tal situación, por lo que no es una función que pueda bloquear el funcionamiento de la red. La cantidad de tiempo de espera es proporcional al largo de los paquetes y a la velocidad de datos, y se configura automáticamente en la etapa de inicialización.

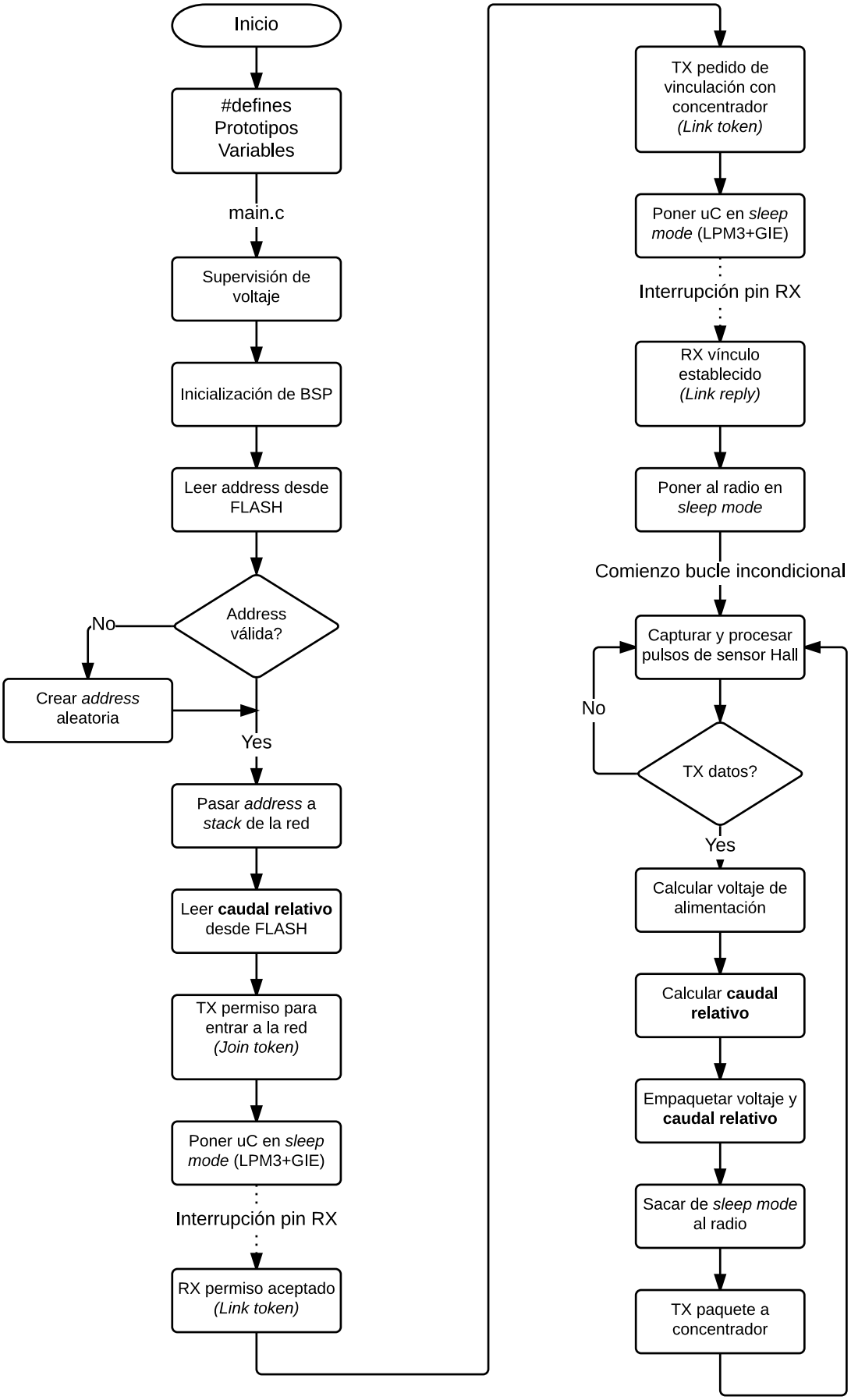


Figura 3.39: Diagrama de flujo del *firmware* del medidor.

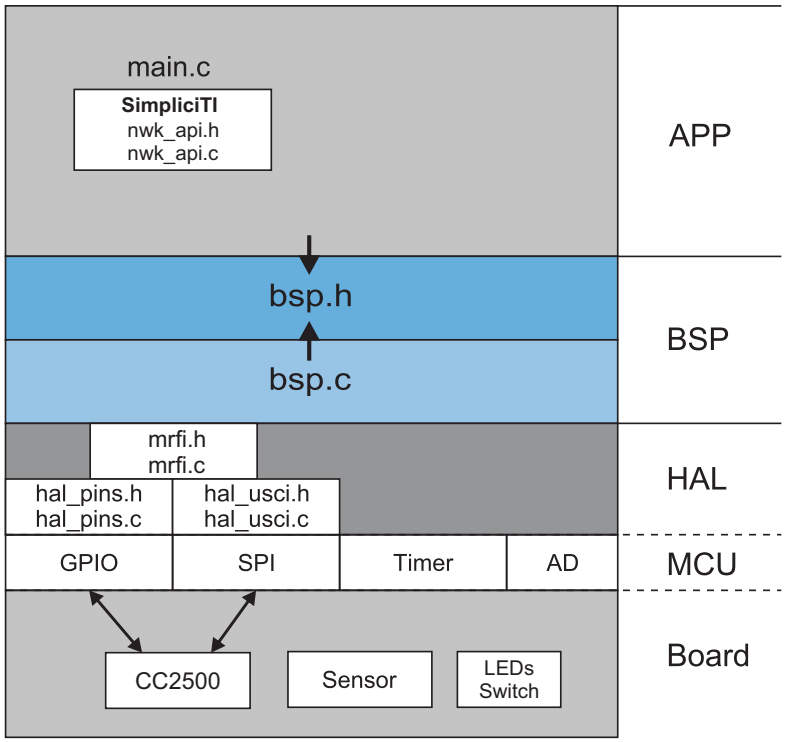


Figura 3.40: Capas del *firmware*.

Esta función puede ser invocada múltiples veces con el objeto de establecer múltiples conexiones.

SMPL_LinkListen(): Esta función pone al nodo en modo de escucha de *links* para vinculación. Estos *links* en *broadcast* son los enviados por los nodos que ejecutan la función `SMPL_Link()`, pidiendo por vinculación. Una vez recibido el *link*, se responde directamente al que lo envió. En el caso de un nodo de tipo *acces-point*, este deberá ejecutar periódicamente esta función con el objeto de agregar *end-devices* a la red.

SMPL_Send(): Esta función envía datos a un nodo, donde el mismo se indica mediante el *link-ID* como parámetro. El código de la red se encarga de manipular el radio durante la transacción. Una vez terminadas las tareas asociadas a esta función el radio vuelve a su situación inicial.

SMPL_Receive(): Esta función chequea el *buffer* de entrada por paquetes recibidos de algún nodo.

A menos que el nodo trabaje en modo *polling* (consulta, en español), esta función no activa el radio o cambia el estado del mismo para recibir. Sólo chequea si ha llegado un paquete en el *link-ID* especificado.

Si el nodo está configurado en modo *polling* en el archivo de configuración del nodo (ver más adelante en **Macros para configuración de los nodos** en esta misma sección), la capa de red se encargará de manejar el radio enviando una consulta y esperando la respuesta asociada.

Si más de un paquete se encuentra en la cola de entrada, el nodo los procesará en orden FIFO (*First Input First Output*). Esto requiere múltiples llamadas de la función `SMPL_Receive()`.

SMPL_Ioctl(): Esta función proporciona una interfaz más potente y detallada con el dispositivo para poder configurarlo u ordenarle acciones específicas como por ejemplo poner en modo *sleep* o bajo consumo al radio, despertarlo del mismo, obtener valores de intensidad de señal, entre otros. Para más detalles de esta función y su funcionamiento consultar la bibliografía oficial del protocolo SimpliciTI [12].

El diagrama de la figura 3.41 muestra un ejemplo de comunicación con un *acces-point* y dos *end-devices* utilizando las funciones explicadas anteriormente.

Como política de acceso al medio, el protocolo SimpliciTI implementa el método «escuchar antes de hablar» (LBT, *Listen Before Talk*) que evita la colisión de paquetes (en los *buffers* de entrada de los *transceivers*) que se transmiten en simultáneo ocupando el mismo espectro de frecuencia. Si el nodo al que se le ordenó transmitir encuentra el medio ocupado, éste reintentará el proceso de chequeo hasta que pueda transmitir. Esto permite una gran efectividad en la recepción de paquetes. En un mismo escenario físico se probaron dos protocolos: SimpliciTI con LBT y otro propietario sin LBT. Con el protocolo sin LBT, la efectividad era del 70 % (en base a 50 transmisiones y visión directa de 20 metros entre nodos), mientras que con SimpliciTI era del 100 % . En ambientes con gran cantidad de *routers* WiFi (que operan en la banda de 2.4 GHz al igual que el CC2500) la efectividad en la recepción disminuía al 40 % al no utilizar LBT. Con SimpliciTI, la efectividad era del 98-100 %.

Macros de SimpliciTI

Para configurar el comportamiento de los nodos y la red existen dos archivos de extensión `.dat` con variables globales o *macros*. El nombre de los archivos y sus correspondientes macros se describen a continuación.

Macros para la configuración de la red

El archivo `smpl_nwk_config.dat` contiene parámetros de configuración de la red en general. Los mismos se describen en la tabla de la figura A.1 en el apéndice A de este informe. Las macros de configuración para la red del sistema y sus valores correspondientes se muestran a continuación. Las sentencias precedidas con el símbolo # se descartan en el proceso de pre-compilación.

`smpl_nwk_config.dat`

```
1 --define=MAX_HOPS=3
2 --define=MAX_HOPS_FROM_AP=1
3 --define=MAX_NWK_PAYLOAD=9
4 --define=MAX_APP_PAYLOAD=10
5 --define=DEFAULT_LINK_TOKEN=0x01020304
6 --define=DEFAULT_JOIN_TOKEN=0x05060708
7 #--define=FREQUENCY_AGILITY
8 #--define=APP_AUTO_ACK
```

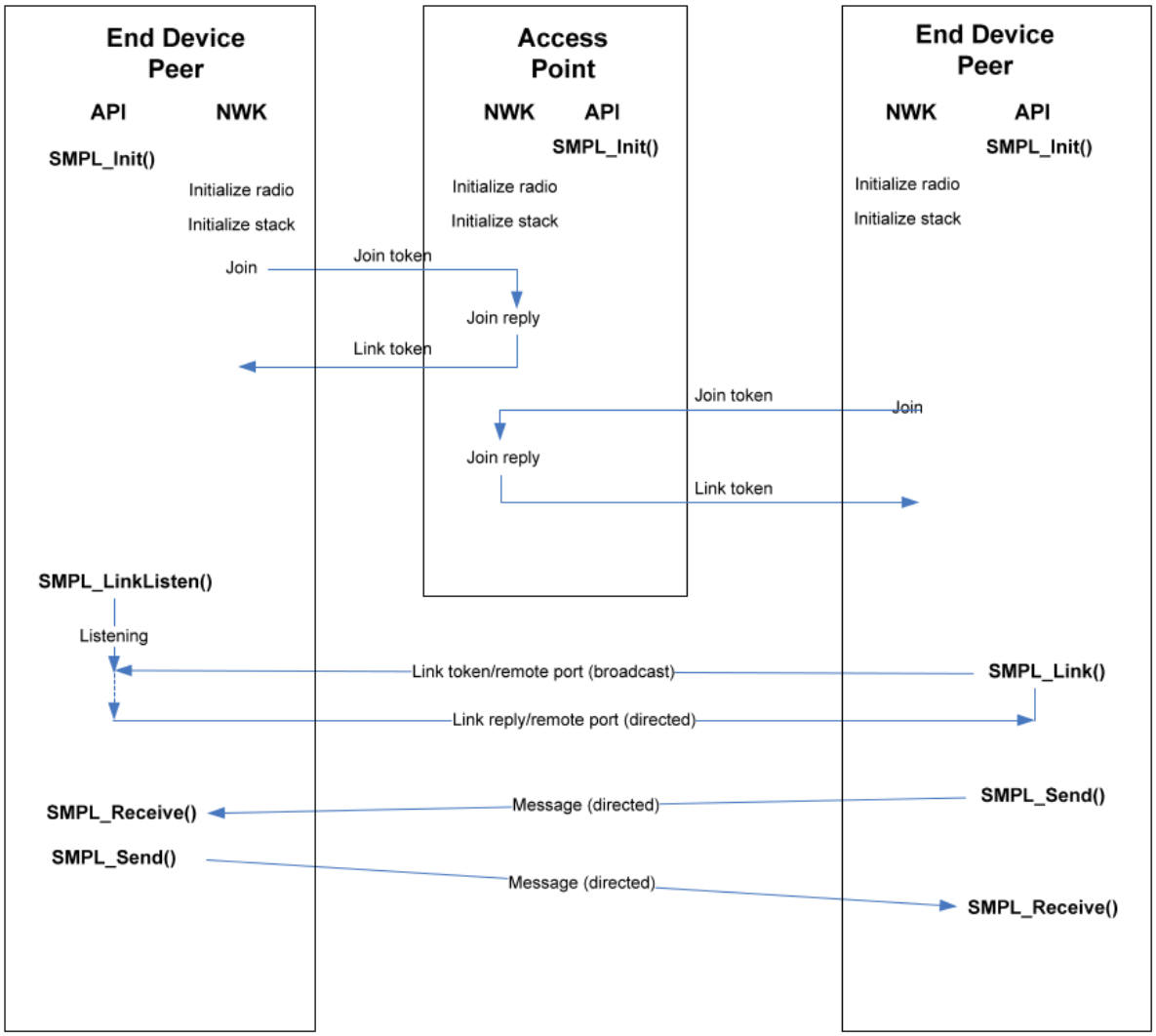


Figura 3.41: SimpliciTI: sesión ejemplo [13].

```
9  --define=EXTENDED_API
10 #--define=SMPL_SECURE
11 #--define=NVOBJECT_SUPPORT
12 --define=SW_TIMER
```

Macros para la configuración de los nodos

El archivo `smpl_config_ED.dat` contiene parámetros de configuración para dispositivos tipo *end-device* como es el caso de los medidores de consumo de este sistema. De igual manera el archivo `smpl_config_AP.dat` contiene parámetros de configuración para dispositivos tipo *access-point* como es el caso del concentrador de datos de este sistema. Los mismos se describen en la tabla de la figura A.2 en el apéndice A de este informe. Las macros de configuración del medidor y sus valores correspondientes se muestran a continuación. Las sentencias precedidas con el símbolo # se descartan en el proceso de pre-compilación.

smpl_config_ED.dat

```
--define=NUM_CONNECTIONS=2
--define=SIZE_INFRAME_Q=2
--define=SIZE_OUTFRAME_Q=2
--define=THIS_DEVICE_ADDRESS="{0x79, 0x56, 0x34, 0x12}"
--define=END_DEVICE
# --define=RX_POLLS
```

3.2.3. Supervisión de voltaje

Según un *erratasheet* proporcionado por Texas Instruments, cuando el *rise time* de la tensión de alimentación es bajo, el microcontrolador entra en un estado en donde el oscilador controlado digitalmente (DCO, *Digitally Controlled Oscillator*) no oscila, y por ende el *firmware* no comienza a ejecutarse. El MCU permanece en estado de *reset* hasta que es desconectado del voltaje. Esta situación ocurría antes de aplicarse un método de supervisión de voltaje. El fabricante aconseja aplicar una tensión de alimentación con un *rise time* mayor o igual a 10 V/s. Como esto no puede lograrse ya que el escenario físico condiciona la tensión de entrada en forma de rampa de bajo *rise time* (figura 3.28) es posible evitar esta situación mediante dos técnicas: con un integrado supervisor de tensión o con un retardo o *delay* al comienzo del programa.

Método con integrado

Se probó en primer lugar un método a nivel de *hardware* mediante el uso de un integrado supervisor de voltaje: el LM8364 de Texas Instruments. Este dispositivo pone al potencial más alto su pin de salida, el cual va conectado al pin MCLR (*Master Clear*) del MCU cuando la tensión del *bus* de alimentación supera los +2.0 V, lo que significa que el microcontrolador permanece en estado de *reset* hasta que el voltaje de entrada del sistema es aceptable.

Aunque el integrado supervisor cumplía correctamente con su tarea, el consumo de corriente de éste se traducía en mayor potencia extraída al generador, significando un aumento de la resistencia de generador de la ecuación 2.4. Esta situación deriva en un mayor flujo necesario para lograr x tensión de salida, lo que se traduce en un corrimiento hacia arriba del flujo mínimo necesario para el arranque del sistema de medición o caudal mínimo detectable. Por el motivo mencionado anteriormente se optó por la segunda opción, que se comenta a continuación.

Método con *delay*

Este método detiene el microcontrolador algunos milisegundos cuando se enciende (momento en que recibe +1.8 V en VCC). Luego del retardo se supone que la alimentación de entrada ya ha dejado de ascender (y además está regulada en +3.3 V) y el MCU comienza a trabajar con normalidad. Esta alternativa no supone un consumo extra, aunque requiere de una calibración manual inicial del retardo para que el tiempo sea el mínimo necesario para evitar la rampa de bajo *rise time*.

3.2.4. Enlace con *Access-Point*

Como se ve en la figura 3.41, el nodo que requiera establecer un vínculo con el concentrador, primero debe ingresar a la red pidiendo el *token* correspondiente con `SMPL_Init()`, y luego establecer un vínculo con el concentrador haciendo `SMPL_Link()`. Al recibir la respuesta, medidor y concentrador quedan vinculados mediante un único *link-ID*.

3.2.5. Modo *sleep* en MCU

Para disminuir el consumo de energía, el microcontrolador y el *transceiver* se configuran para operar en estado inactivo o modo *sleep*. En el diagrama de flujo de la figura 3.39 se observa en una de las etapas el ingreso a LMP3 (*Low Power Mode 3*) por parte del MCU. Esto significa que el CPU, el reloj principal MCLK (*Main Clock*), el oscilador secundario SMCLK (*Sub-main Clock*) y el generador de corriente contínua (DC, *Direct Current*) para el oscilador DCO permanecen desactivados, mientras que solamente permanece activado el reloj auxiliar ACLK (*Auxiliary Clock*). En modo LPM3, el microcontrolador consume entre 0.9 μA y 1.5 μA con 3 V en VCC.

3.2.6. Lectura y tratamiento de pulsos externos

Los pulsos generados por el sensor de efecto Hall ingresan en el pin 2.0 del microcontrolador. El mismo contabiliza pulsos ascendentes. Una revolución del rotor seco equivale a un pulso detectado o *tick*. Se creó una variable `tx_after_ticks` para determinar la cantidad de *ticks* necesarios para realizar «una» transmisión. Esta variable permite ajustar la cantidad de agua que dispara una transmisión. Se muestra a continuación la sección de código que lee y procesa los pulsos de entrada.

```
1      if (!(P2IN & 0x01) && !ctrl_sw) {
2        //BSP_TURN_ON_LED1();
3        tx_after_ticks--;
4        ctrl_sw = 1;
5      } else if ((P2IN & 0x01) && ctrl_sw) {
6        //BSP_TURN_OFF_LED1();
7        ctrl_sw = 0;
8      }
9
10     if (tx_after_ticks == 0) {
11       sSelfMeasureSem = 1;
12       tx_after_ticks = TX_AFTER_TICKS;
13     }
```

El manejo de `LED1` se muestra comentado (ignorado en el proceso de pre-compilación) ya que sólo se utilizó durante el *debugging* del sistema. Cuando se realiza una transmisión desde el medidor al concentrador, lo que se envía en relación a la lectura de caudal es un valor de «caudal relativo». El caudal relativo es un entero que varía entre 1 y 100 con incrementos de 1 por cada transmisión efectuada. Si un paquete enviado por el medidor no llega al concentrador, este último podrá identificar aquel paquete perdido y considerarlo en el caudal total. Una diferencia mayor a uno entre el último caudal relativo recibido y

el anterior inmediato, supone paquetes perdidos entre ambas recepciones. Por ejemplo, si el último valor de caudal relativo recibido por el concentrador corresponde a 20 y el valor anterior inmediato fue de 17, entonces, el concentrador no ha recibido correctamente dos paquetes de caudal relativo (17 si, 18 no, 19 no, 20 si: dos paquetes perdidos). El encargado de detectar y corregir estos errores es el concentrador de datos. Esta metodología evita perder lecturas de caudal debido a posibles fallas durante la transmisión.

3.2.7. Almacenamiento no volátil de caudal relativo

Se creó la variable `save_data_after_txs` para configurar el intervalo de almacenamiento no volátil en la memoria flash del microcontrolador del caudal relativo. Esta variable define cada cuántas transmisiones de caudal relativo se almacena un valor en la memoria no volátil del MCU. Así como se hace con la dirección propia del nodo, el último caudal relativo es almacenado en la flash cada `save_data_after_txs` transmisiones. Esta práctica evita que el último valor de caudal relativo se borre cuando el microcontrolador pierde su alimentación entre cada sesión.

```
1      /* Flow variable flash writing control */
2      save_data_after_txs--;
3      if (save_data_after_txs == 0) {
4        writeFlashFlow();
5        save_data_after_txs = SAVE_DATA_AFTER_TXS;
6      }
```

Capítulo 4

El Concentrador de Datos

4.1. *Hardware*

El *hardware* correspondiente al medidor se representa con el diagrama de bloques de la figura 4.1. Según el método comentado en la sección 2.4, cada bloque será dividido en cuatro etapas con el fin de abarcar todo el material que se necesita exponer. Se puede ver al diagrama de bloques del *hardware* del concentrador de datos como un índice gráfico del contenido de esta sección.

Los bloques de *hardware* **MCU—microcontrolador**, ***transceiver* y adaptación de antena** y **LED para *debugging*** correspondientes al concentrador son exactamente iguales en diseño y fabricación a los bloques expuestos en las secciones 3.1.4, 3.1.5 y 3.1.6 respectivamente.

4.1.1. Interfaz USB

Concentrador.InterfazUSB.*IdeaDesign*

El microcontrolador envía los paquetes con caudales relativos a la base de datos a través de su módulo UART (*Universal Asynchronous Receiver-Transmitter*). Sin embargo, la base de datos solo puede recibir los paquetes a través de uno de los puertos USB

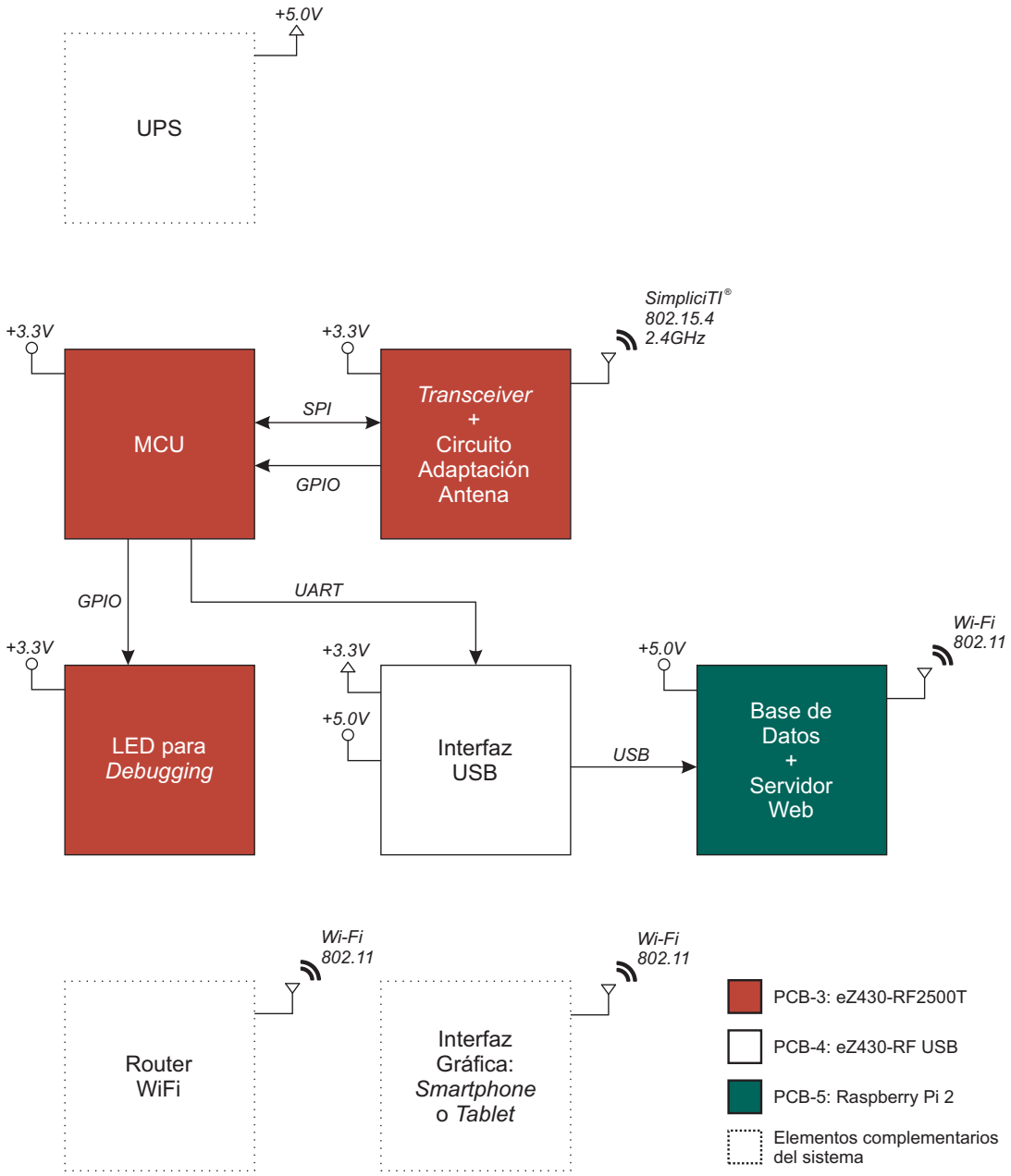
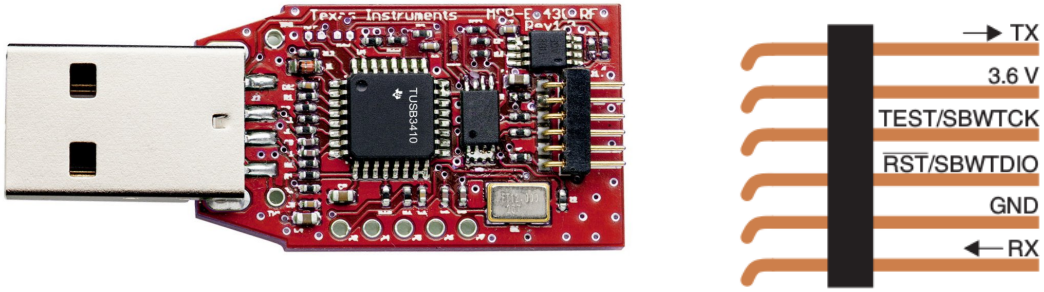


Figura 4.1: Diagrama de bloques del *hardware* del concentrador.

del PCB número cinco del sistema (ver figura 4.1). Para enlazar entonces PCB-3 y PCB-5 (salida UART y entrada USB) es necesario un bloque intermedio que adapte ambos protocolos.

Concentrador.InterfazUSB.*PartsOnHand*

Como solución se utilizó la interfaz *eZ430-RF USB* que se muestra en la figura 4.2. Este dispositivo es parte del kit de desarrollo eZ430-RF2500. Su función es programar el microcontrolador en la etapa de desarrollo y servir de interfaz UART-USB entre microcontrolador y dispositivo con conexión USB.



(a) eZ430-RF USB. (b) Pines de conexión.

Figura 4.2: Interfaz USB para programación/*debugging* y UART.

Concentrador.InterfazUSB.*Assembling*

Debido a que el adaptador eZ430-RF USB pertenece al kit eZ430-RF2500, puede conectarse fácilmente al PCB que aloja al microcontrolador. En la figura 4.3 vemos ambos PCBs conectados por medio de los pines de conexión de la figura 4.2b.

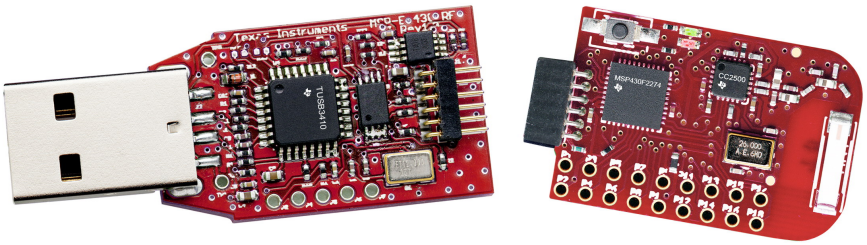


Figura 4.3: eZ430-RF2500 con interfaz UART-USB eZ430-RF USB.

4.1.2. Base de datos y servidor Web

Concentrador.WebServer.*IdeaDesign*

La información proporcionada por los medidores de la red y recolectada por el concentrador debe ser post-procesada para que pueda ser correctamente almacenada en una base de datos y posteriormente ofrecida cuándo y cómo el usuario lo desee. Este post-procesamiento implica corrección de errores en caudales relativos asociados a paquetes no recibidos, como también diversificar y empaquetar la información para que pueda ser almacenada ordenadamente en una base de datos. Como último eslabón se encuentra la visualización de la información recolectada y procesada. Para ello, se seleccionó como mejor opción una aplicación Web multi-plataforma que pueda acceder a la base de datos en tiempo real.

Para lograr todo lo anterior, es necesario un *hardware* con capacidades acordes a los requerimientos de procesamiento, almacenamiento y conectividad.

Concentrador.WebServer.*PartsOnHand*

Luego de considerar distintas alternativas de *hardware* con la *performance* que la idea requiere, se eligió una mini-computadora que permite embeber un sistema operativo para la ejecución de tareas, muchas de ellas programadas en distintos lenguajes y entornos. Este dispositivo se trata del *Raspberry Pi 2*. La computadora del tamaño de una tarjeta de crédito se muestra en la figura 4.4.

Las características principales de este dispositivo se indican en la siguiente lista:

- 900 MHz quad-core ARM Cortex-A7 CPU
- 1 GB RAM
- 4 puertos USB
- 40 pines de propósito general (GPIO, *General Pourpose I/O*)
- Full HDMI
- Puerto Ethernet
- *Jack* 3.5 mm audio combinado con video compuesto
- Interfaz para cámara (CSI)
- Interfaz para *display* (DSI)
- *Slot* para tarjeta micro SD
- *Core* de gráficos 3D VideoCore IV

Debido al hecho de contar con un procesador ARMv7, este dispositivo puede *correr* todas las distribuciones de ARM GNU/Linux como así también Microsoft Windows 10.

La mini-computadora tiene un precio de $990 en Argentina. Su valor en Estados Unidos es de USD 25.

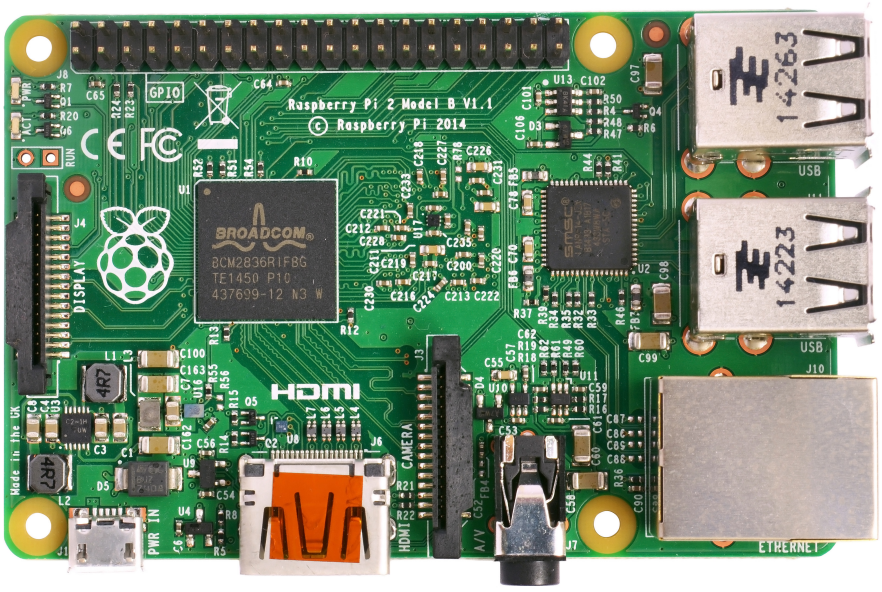


Figura 4.4: Mini-computadora Raspberry Pi 2 Model B.

Concentrador.WebServer.*Assembling*

El *access-point* y el servidor Web se conectan a través de la interfaz UART-USB mencionada en la sección 4.1.1. El concentrador completo se muestra en la figura 4.5.

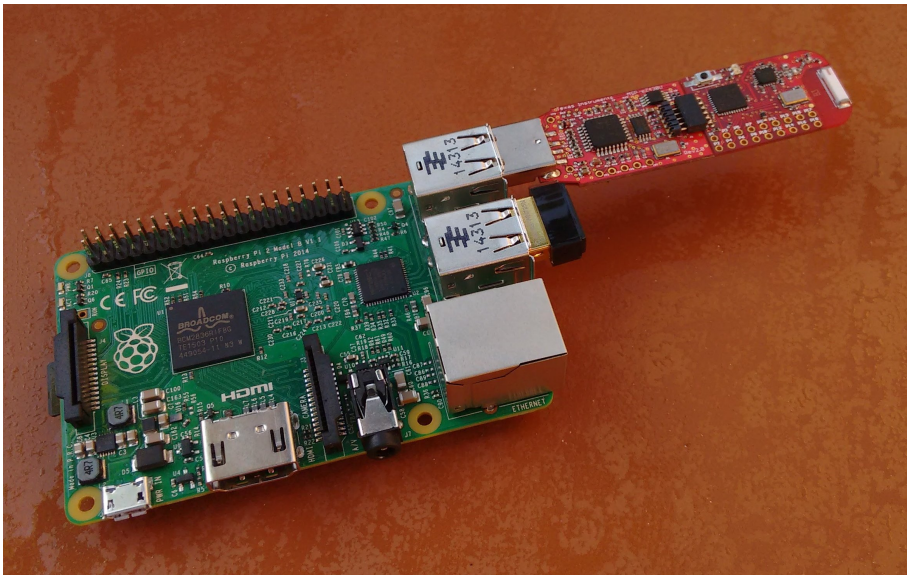


Figura 4.5: Concentrador de datos.

4.2. *Firmware*

En esta sección se presenta el *firmware* del concentrador de datos. Parte del *firmware* o programa se graba y ejecuta en el microcontrolador presentado en la sección 3.1.4, mientras que otra parte se graba y se ejecuta en la mini-computadora presentada en la sección 4.1.2. El funcionamiento de tales programas se describe en el diagrama de flujo de la figura 4.6.

4.2.1. Inicialización de la red

Cuando el *access-point* inicializa la red debe pasar como parámetro a la función de inicialización el nombre de la función que se llamará cuando el concentrador reciba un paquete. En este caso la inicialización queda `SMPL_Init(sCB)`, donde `sCB` (*simpliciTI Callback*) es la función a ejecutarse. Esta función siempre se ejecuta en contexto de servicios de interrupción (ISR, *Interrupt Service Routine*).

```
1    SMPL_Init(sCB);
```

4.2.2. Recepción y procesamiento de paquetes

***Callback* de recepción**

```
1  static uint8_t sCB(linkID_t lid)
2  {
3    if (lid)
4    {
5      sPeerFrameSem++;
6      sBlinky = 0;
7    }
8    else
9    {
10     sJoinSem++;
11   }
12
13   /* leave frame to be read by application. */
14   return 0;
15 }
```

Observar en la función `sCB()` que si el parámetro `lid` es distinto de cero se incrementa la variable `sPeerFrameSem`. Esto significa que el identificador de vínculo `lid` ya se ha creado para el nodo que envió el paquete, lo que significa una transacción de datos y no un pedido de *link token* para unirse a la red. Para este último caso, la variable que se incrementa es `sJoinSem`. Estas variables se utilizan como semáforos de *software* para ejecutar tareas correspondientes fuera del contexto de interrupción, en la aplicación principal.

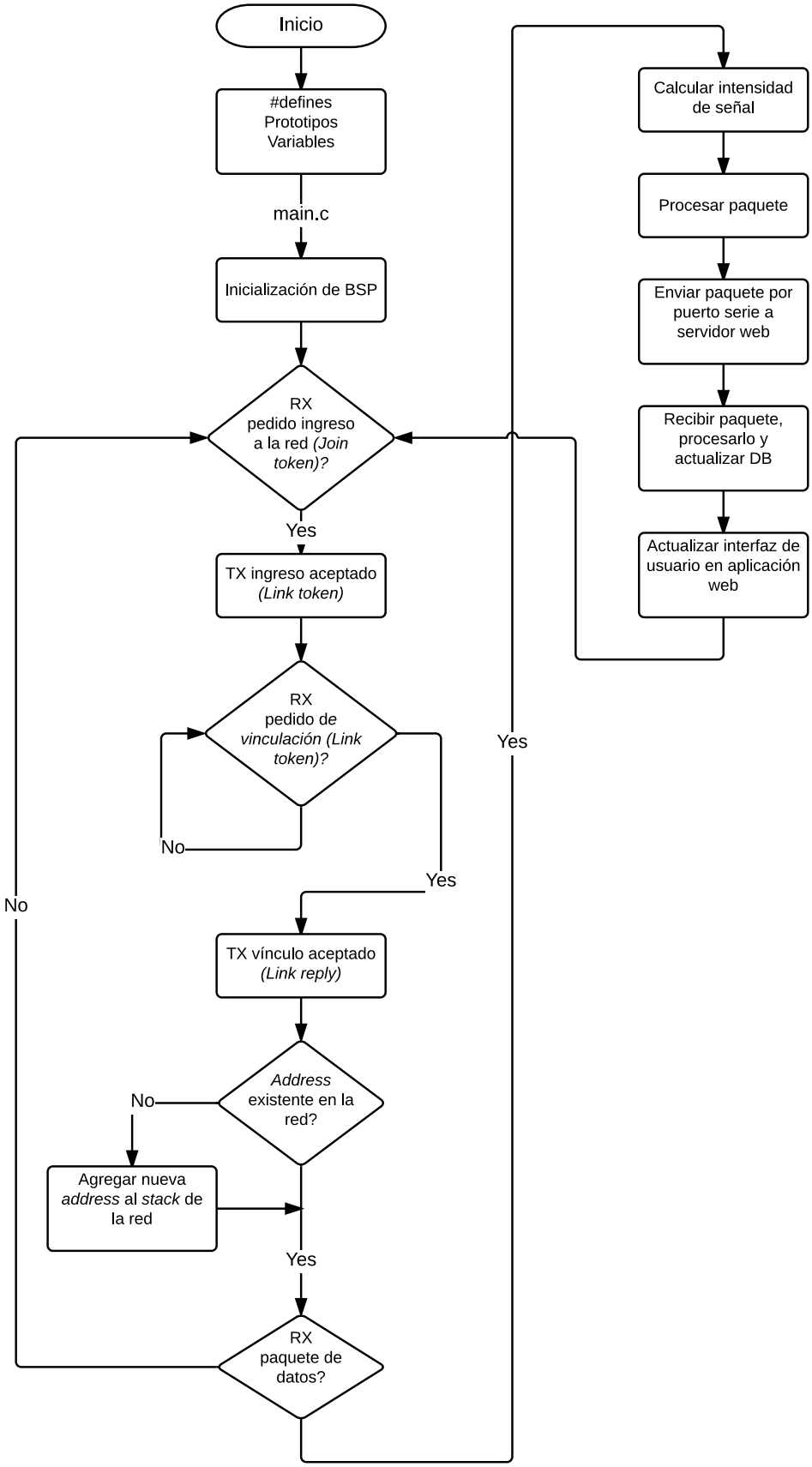


Figura 4.6: Diagrama de flujo del *firmware* del concentrador.

Procesamiento de `sPeerFrameSem`

Cuando el semáforo `sPeerFrameSem` es incrementado por el *callback* de recepción, se debe atender el o los paquetes que han ingresado al *buffer* de entrada del *transceiver* con la función `SMPL_Receive()`, presentada en la sección 3.2.2.

```
1 if (SMPL_SUCCESS == SMPL_Receive(sLID[i], msg, &len)) {
2   ioctlRadioSiginfo_t sigInfo;
3   processMessage(sLID[i], msg, len);
4   sigInfo.lid = sLID[i];
5   SMPL_Ioctl(IOCTL_OBJ_RADIO, IOCTL_ACT_RADIO_SIGINFO, (void *)&sigInfo);
6   transmitData( i, sigInfo.sigInfo.rssi, (char*)msg );
```

La carga útil recibida se transfiere desde la FIFO de entrada del *transceiver* a un espacio de memoria en el MCU representado por `msg`. Observar en el trozo de código anterior que una vez realizada la transferencia con `SMPL_Receive()` se consulta el identificador del *link* (`sLID[i]`) y la intensidad de señal con la que se recibió el paquete (`sigInfo`) y se procesa `msg` con la función `processMessage()`. Estos tres elementos se transmiten finalmente por la UART del microcontrolador. En este caso el procesamiento de `msg` sólo está limitado a encender LEDs de *debugging* para indicar recepción de paquetes. El verdadero procesamiento de la información (y, sobre todo, del caudal relativo) se da en el motor de interpretación y almacenamiento de datos creado con Python y ejecutado en la mini-computadora. Se detalla en la sección 4.2.5.

4.2.3. Raspbian: RaspberryPi + Debian Linux

Raspbian es una distribución del sistema operativo GNU/Linux y por lo tanto libre basado en Debian Wheezy (Debian 7.0) para la placa computadora Raspberry Pi.

Técnicamente el sistema operativo es un *port* no oficial de Debian Wheezy armhf para el procesador (CPU) de Raspberry Pi, con soporte optimizado para cálculos en coma flotante por *hardware*, lo que permite dar más rendimiento en según que casos. El *port* fue necesario al no haber versión Debian Wheezy armhf para la CPU ARMv7 que contiene el Raspberry PI.

La distribución usa LXDE como escritorio (figura 4.7) y Midori como navegador Web. Además contiene herramientas de desarrollo como IDLE para el lenguaje de programación Python.

Sobre este sistema operativo se ejecuta el motor de interpretación y almacenamiento de datos programado con Python. En simultáneo, se ejecuta la aplicación Web de tiempo real programada en Meteor. Además, su sistema de archivos almacena la base de datos no-SQL MongoDB y ejecuta el *driver* correspondiente para su acceso. Actuando juntos sobre Raspbian, estos elementos conforman los módulos *Actualizar DB y Aplicación Web* del diagrama de la figura 4.6.

Figura 4.7: *Screenshot* de Raspbian.

4.2.4. Introducción a MongoDB

MongoDB es un sistema de base de datos NoSQL orientado a documentos, desarrollado bajo el concepto de código abierto.

MongoDB forma parte de la nueva familia de sistemas de base de datos NoSQL. En vez de guardar los datos en tablas como se hace en las base de datos relacionales, MongoDB guarda estructuras de datos en documentos tipo JSON con un esquema dinámico (MongoDB llama ese formato BSON), haciendo que la integración de los datos en ciertas aplicaciones sea más fácil y rápida.

Casos de uso

- Almacenamiento y registro de eventos
- Sistemas de manejo de documentos y contenido
- Comercio Electrónico
- Juegos
- Problemas de alto volumen de lecturas

- Aplicaciones móviles
- Almacén de datos operacional de una página Web
- Manejo de contenido
- Almacenamiento de comentarios
  - Votaciones
  - Registro de usuarios
  - Perfiles de usuarios
  - Sesiones de datos
  - etc.
- Proyectos que utilizan metodologías de desarrollo iterativo o ágiles
- Manejo de estadísticas en tiempo real

Manipulación de datos: colecciones y documentos

MongoDB guarda la estructura de los datos en documentos tipo JSON con un esquema dinámico llamado BSON, lo que implica que no existe un esquema predefinido. Los elementos de los datos se denominan documentos y se guardan en colecciones.

Una colección puede tener un número indeterminado de documentos. Comparando con una base de datos relacional, se puede decir que las colecciones son como tablas y los documentos son registros en la tabla. La diferencia es que en una base de datos relacional cada registro en una tabla tiene la misma cantidad de campos, mientras que en MongoDB cada documento en una colección puede tener diferentes campos. En un documento, se pueden agregar, eliminar, modificar o renombrar nuevos campos en cualquier momento, ya que no hay un esquema predefinido.

La estructura de un documento es simple y está compuesta por *key-value pairs*, esto es debido a que MongoDB sigue el formato de JSON. En MongoDB la clave o *key* es el nombre del campo y el valor o *value* es su contenido, los cuales se separan mediante el uso de ":", tal y como se puede ver en el siguiente ejemplo. Como valor se pueden usar números, cadenas o datos binarios como imágenes o cualquier otro *key-value pair*.

```
{
    "_id": ObjectId("4efa8d2b7d284dad101e4bc7"),
    "Node": "01",
    "RelativeFlow": 45,
    "TotalFlow": 3659,
    "Battery": "3.1V",
    "Strength": "38%",
    "RE": "no"
}
```

MongoDB es el sistema de base de datos adoptado de manera oficial por Meteor, el *framework* usado para crear la aplicación Web de tiempo real. Cuando se inicializa un

proyecto con Meteor, éste automáticamente crea una base de datos MongoDB asociada únicamente a la aplicación. En la sección 4.2.5 se explica de qué manera se accede y modifica esa base de datos.

4.2.5. Post-procesamiento con Python

Python es un lenguaje de programación simple y flexible con gran potencialidad gracias a su amplia variedad de extensiones. Estas extensiones permiten vincular fácilmente la propia aplicación python con elementos externos como otro *software* embebido o *hardware* del dispositivo que ejecuta la aplicación.

La aplicación diseñada con python se ejecuta en la mini-computadora con Raspbian. Esta aplicación se encarga de recolectar los paquetes que entran por el puerto USB de Raspberry, decodificarlos, convertirlos a formato JSON, procesar el valor de caudal relativo y almacenar en base de datos.

Pyserial: lectura de puerto USB

Con la extensión PySerial para Python se puede establecer conexión con el puerto USB de la placa Raspberry Pi. Para incluir la extensión se debe importar la misma de la siguiente forma:

```
1 import serial
```

Para establecer conexión es necesario conocer la ruta específica hacia el puerto que está mapeado en memoria por el sistema operativo. En este caso, la ruta hacia el puerto usado bajo Linux Raspbian es `/dev/ttyACM0`.

Para recolectar los paquetes provenientes de la UART del MCU se diseñó una rutina que intenta conectar al puerto y repite esta acción hasta que tal conexión sea exitosa. Para lograr este comportamiento de reintento la herramienta `try` evalúa la función `serial` con el objetivo de encontrar excepciones si la conexión no puede establecerse. Una vez lograda la conexión, se imprime en la pantalla de *debugging* el nombre del enlace activo.

```
1  # Conectarse al puerto serie
2  serialConnected = False
3  print('Buscando puerto... ')
4  while not serialConnected:
5      try:
6          ser = serial.Serial('/dev/ttyACM0', 9600)
7          serialConnected = True
8      except serial.SerialException:
9          pass
10
```

```
11 print(ser.name)
12 print('----------------------------------------------------')
```

Decodificación de paquetes

En la figura 4.8 se observa una secuencia de líneas con cinco campos: *Node* (ID del medidor), *Flow* (caudal relativo), *Battery* (voltaje del *bus* de alimentación), *Strength* (intensidad de señal) y *RE* (saltos del paquete), y sus respectivos valores. Cada línea de la figura corresponde a un paquete recibido por la aplicación python. La misma debe encargarse de identificar y seccionar los pares clave-valor (*key-value pairs*) para poder convertirlos en formato de *diccionario python*. Este formato es especial para trabajar con *key-value pairs* ya que los almacena como tales dentro de un misma variable, pudiendo acceder y modificar fácilmente campos y valores específicos dentro de ella.

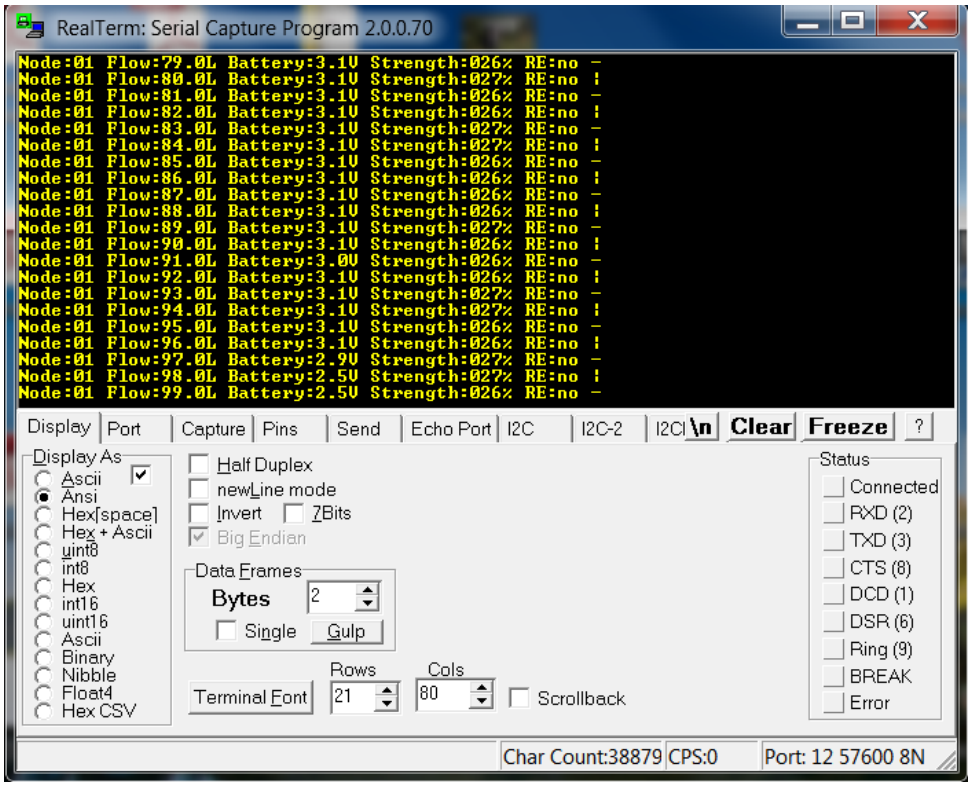


Figura 4.8: Paquetes recibidos y procesados por Python

En las sentencias que siguen a continuación se leen los paquetes y se convierten a diccionarios python.

```
1 while True:
2     # read, decode binary to string and remove whitespace characters
3     line = ser.readline().decode('ascii').strip()
4 
5     if line == 'ED Joined':
6         print(line)
7     else:
```

```
8           # convert string to python dictionary (key/value pairs)
9           key_value = dict(u.split(":") for u in line.split(","))
```

La sentencia de la línea 9 transforma la cadena de caracteres *ASCII* en un diccionario python con los pares clave-valor que la cadena contenga. Es importante notar que todo eso es realizado en una sola sentencia gracias a la posibilidad que otorga python de concatenar funciones incluso dentro de la zona de parámetros de una función.

En este caso, si se observa la sentencia de derecha a izquierda, se crea un *array* de *strings* de nombre `u` a partir del *string* principal que es seccionado por la función `split()`. La misma toma como parámetro el carácter que establece la separación entre *strings*: la coma. Luego, en la misma sentencia, se ejecuta una secuencia `for` sobre el *array* `u` en donde en cada ciclo, cada elemento de `u` se reorganiza en pares *key-value* con `split(":")`. Estos valores *key-value* son los parámetros de la función `dict()` que crea un diccionario python por cada paquete de entrada y lo llama `key_value`.

Pymongo: *driver* Python para MongoDB

Antes de continuar con el tratamiento de los paquetes recibidos es necesario presentar Pymongo, el *driver* de Python para el manejo de la base de datos MongoDB. Pymongo es el *driver* Python recomendado en el sitio oficial de MongoDB. Esta extensión se importa de la siguiente manera:

```
1 import pymongo
2 from pymongo import MongoClient
```

Esta extensión permite comunicarse con la base de datos MongoDB transformándonos en *clientes* de la misma al pasar como parámetro de la función `MongoClient()` la ubicación de dicha base de datos. De la misma forma que se realizó con la conexión al puerto USB, se diseñó una rutina de reintento para establecer conexión con MongoDB. La base de datos sólo está disponible una vez inicializado Meteor (se detalla más adelante).

```
1  # Conectarse con la mongoDB de Meteor (running)
2  print('-------------------------------------------------')
3  print('Conectando a MongoDB...   ')
4
5  mongoConnected = False
6  while not mongoConnected:
7      try:
8          client = MongoClient('mongodb://127.0.0.1:3001/meteor')
9          mongoConnected = True
10     except pymongo.errors.ConnectionFailure:
11         pass
12
13 print('mongo running detected!')
14 print(client)
15 databases = str(client.database_names())
```

```
16 print('DBs disponibles: ' + databases)
17 db = client.meteor
18
19 collectionsAvailable = False
20 while not collectionsAvailable:
21     try:
22         collections = str(db.collection_names())
23         collectionsAvailable = True
24     except pymongo.errors.AutoReconnect:
25         pass
26
27 print('Colecciones disponibles: ' + collections)
28 print('-------------------------------------------------')
```

Para más información sobre el conjunto completo de funciones de Pymongo consultar su API en su página Web oficial [14].

Pull de medidores

Cada vez que el concentrador es encendido (o reiniciado por alguna motivo particular) recupera las direcciones de los nodos de la red (los medidores) desde la base de datos. Estos medidores han sido dados de alta con anterioridad. Esta recuperación se hace para que el proceso de alta de nuevo nodo en base de datos no se repita para nodos ya agregados. Si el nodo ya existe, se modifican sus campos existentes. El trozo que corresponde al *pull* de nodos se muestra a continuación:

```
1 # Recovery nodes' address in "nodesInMongo" list
2 nodesInMongo = []
3 cursor = db.sensors.find({}, {"node": 1, "_id": 0})
4 for record in cursor:
5     nodesInMongo.append(record["node"])
6 print('Nodes in mongo:', nodesInMongo)
```

Observar en la línea 3 del código anterior que con la función `.find()` de Pymongo se obtiene un cursor apuntando al resultado, esto es, una especie de puntero al *array* de resultados que coinciden con el *match criteria* otorgado como parámetro.

Procesamiento de caudal relativo

Si el nodo cuyo caudal debe procesarse existe en la lista creada en la etapa de *pull*, entonces se consulta en base de datos su último valor de caudal relativo (`ofid`—*old flow id*) y se calcula el incremento en función del nuevo caudal relativo. El cálculo es simplemente el caudal relativo actual menos el caudal relativo anterior registrado en base de datos (línea 11 del código a continuación).

Si el nodo no existe en la lista de nodos registrados, entonces se inserta un nuevo documento (correspondiente al nuevo nodo) con valor de caudal total cero y se agrega el medidor a la lista.

Por último, se actualizan los documentos de los medidores que han registrado incrementos en su caudal total. Esta acción se lleva a cabo en las líneas 21 y 22 del código mostrado a continuación.

```
1  # convert flow value (string) to int to manipulate after
2  key_value['flow'] = int(key_value['flow'])
3
4  nodeAddrInput = key_value.get('node')
5  nodeFlowInput = key_value.get('flow')
6
7  if nodeAddrInput in nodesInMongo:
8    old = db.sensors.find_one({"node": nodeAddrInput},
9    {"ofid": 1, "_id": 0}) #ofid: old flow id
10   old = old["ofid"]
11   flowToInc = nodeFlowInput - old
12   if flowToInc < 0:
13     flowToInc = 1
14 else:
15   db.sensors.insert({"node": nodeAddrInput, "flow": 0,
16   "ofid": nodeFlowInput})
17   nodesInMongo.append(nodeAddrInput)
18   flowToInc = 1
19
20 oldFlows[nodeAddrInput] = nodeFlowInput
21 db.sensors.update({"node": nodeAddrInput},
22 {"$inc": {"flow": flowToInc}, "$set": {"ofid": nodeFlowInput}})
23 print(key_value)
```

Notar que mediante el comando `$inc` se suma `flowToInc` al campo `flow`. En cambio con `$set`, se sobreescribe el campo `ofid` con el valor `nodeFlowInput`.

4.2.6. Meteor: aplicación Web de tiempo real

Introducción

Meteor es una plataforma para desarrollar aplicaciones Web o WAF (*Web Application Framework*) JavaScript *open-source* (código abierto) desarrollado sobre Node.js. Meteor permite una rápida creación de prototipos y produce código multi-plataforma (Web, Android, iOS). Integra de manera nativa MongoDB como base de datos, usa un protocolo de datos distribuidos (DDP, *Distributed Data Protocol*) y un comportamiento *publish-subscribe* (publicación-subscripción) para propagar automáticamente a los clientes cambios en los datos sin la necesidad de que el desarrollador escriba código de sincronización. Del lado del cliente, Meteor utiliza jQuery y puede ser usado con cualquier librería de interfaz de usuario JavaScript.

Definición de WAF

Un WAF es un entorno de *software* diseñado para hacer más liviano y simple el desarrollo de sitios Web dinámicos, aplicaciones Web, servicios Web y recursos Web. El *framework* o plataforma ayuda a aliviar la carga asociada con actividades comunes relacionadas al desarrollo Web. Por ejemplo, muchos entornos de este tipo proveen librerías para acceder a bases de datos, uso de plantillas para la visual y administración de sesión de usuarios.

Definición de Node.js

Node.js es un entorno multi-plataforma que sirve para desarrollar aplicaciones Web que se ejecutan del lado del servidor. Las aplicaciones Node.js son escritas en JavaScript y pueden correr en OS X, Microsoft Windows, Linux, FreeBSD, NonStop, IBM AIX, IBM System z e IBM i.

Node.js provee una arquitectura orientada a eventos y una *non-blocking I/O API* diseñada para optimizar el *throughput* (tasa de envíos exitosos dentro de un canal de comunicación) y la escalabilidad para aplicaciones Web de tiempo real. Utiliza Google V8 como motor para ejecutar JavaScript. Node.js incluye una librería que permite a las aplicaciones actuar como servidores Web sin la necesidad de hacerlo a través de Apache HTTP Server, Nginx o IIS.

Definición de DDP

DDP o protocolo de datos distribuidos es un protocolo cliente-servidor que sirve para consultar y actualizar una base de datos en el servidor y sincronizar tales actualizaciones entre los clientes. Utiliza el sistema de mensajería publicación-suscripción. Este protocolo fue creado para ser usado por Meteor.

Se puede pensar en el sistema de publicación-suscripción como un embudo que transfiere los datos desde una colección en el servidor (origen) a una en un cliente (destino). El protocolo que se habla dentro del embudo se llama DDP.

El funcionamiento de Meteor

Como se mencionó anteriormente, Meteor es una plataforma para crear aplicaciones Web en tiempo real construida sobre Node.js. Meteor se localiza entre la base de datos de la aplicación y su interfaz de usuario y asegura de que ambas partes estén sincronizadas (figura 4.9).

Como Meteor usa Node.js, se utiliza JavaScript en el cliente y en el servidor. Meteor es capaz de compartir código entre ambos entornos.

El resultado es una plataforma muy potente y muy sencilla ya que Meteor abstrae muchas de las molestias y dificultades que se encuentran habitualmente en el desarrollo de aplicaciones Web.

La principal innovación de Meteor es que, mientras que una aplicación *Rails* solo vive en el servidor, una aplicación Meteor también incluye componentes que se ejecutarán en el cliente (el navegador). Lo más importante es lo que Meteor llama *base de datos en*

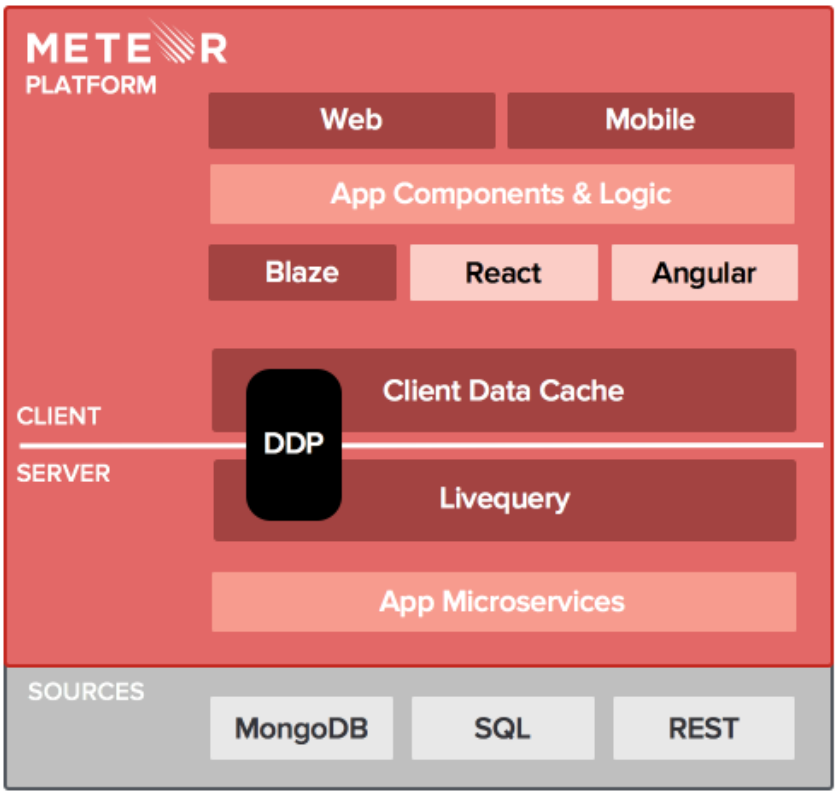


Figura 4.9: Plataforma Meteor y recursos [15].

todas partes. En pocas palabras, Meteor tomará un subconjunto de la base de datos y la *copiará* en el cliente (figura 4.10). Esto tiene dos grandes implicaciones: la primera es que en lugar de enviar código HTML (*HyperText Markup Languaje*), una aplicación Meteor envía datos en bruto al cliente y deja que el cliente se ocupe de ellos (*data on the wire*). Lo segundo es que el cliente será capaz de acceder, e incluso modificar esos datos de manera instantánea sin tener que esperar al servidor (*latency compensation*).

La aplicación Meteor de WiWater

A la hora de instalar Meteor en Raspbian se presentó un inconveniente: Meteor 1.1 no puede ser ejecutado sobre arquitecturas ARM, la que posee el microprocesador de RaspberryPi. Sin embargo, gracias al hecho de ser de código abierto, existe un *fork* del proyecto Meteor que ha sido desarrollado por el equipo de UDOO (www.udoo.org) permitiendo ejecutar Meteor en dispositivos ARM. Un tutorial de los pasos a seguir para instalar Meteor y MongoDB en Raspbian (arquitectura ARM) se encuentra en el sitio Web especificado en la bibliografía [16].

El archivo HTML

En el archivo de extensión `.html` se encuentra el contenido de la aplicación. El usuario va a interactuar con ese contenido, por lo que debe ser simple e intuitivo. Ésta es una parte muy importante ya que el éxito o el fracaso del sistema completo también está relacionado a una buena o mala experiencia de usuario.

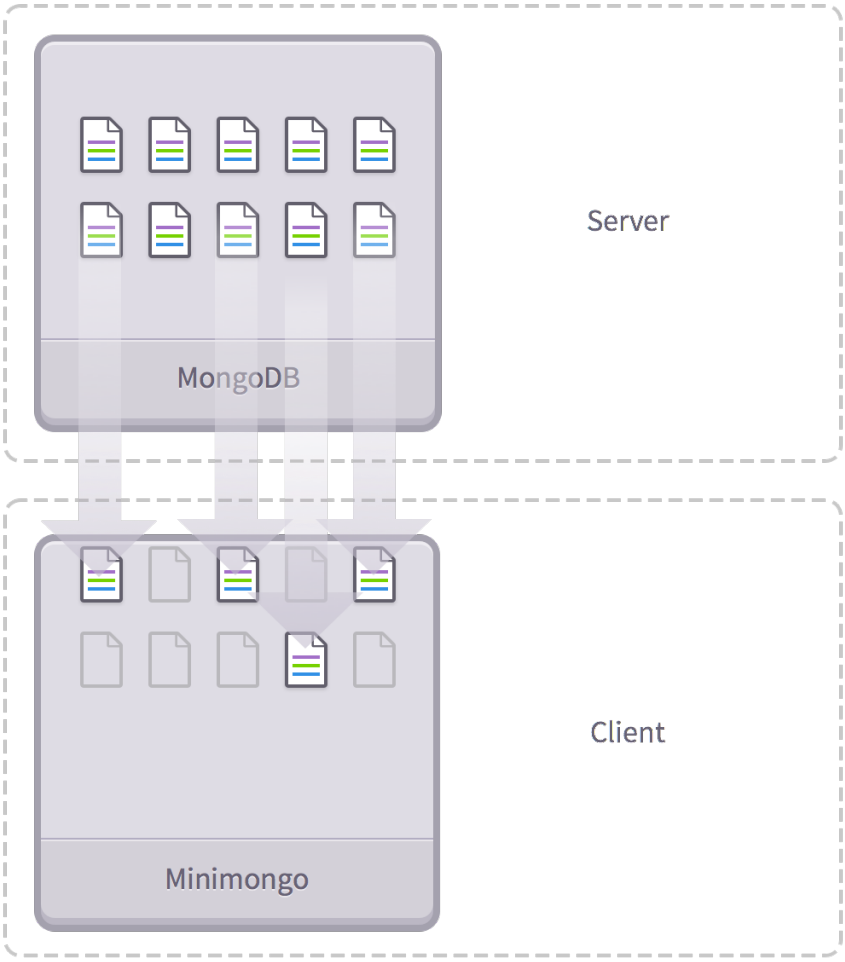


Figura 4.10: Subconjunto de la base de datos en el cliente [15].

El HTML se compone de tres partes: una cabecera o `<head>`, un cuerpo o `<body>` y la zona de definición de *templates*. A diferencia de un HTML convencional, un HTML enmarcado en un proyecto Meteor no necesita de elementos extras. No obstante, el motor que compila el proyecto agrega lo necesario al HTML para que se ejecute con normalidad en cualquier navegador. Ésta es otra característica de Meteor que simplifica la tarea del desarrollador.

El archivo JavaScript

En el archivo de extensión `.js` se encuentran los componentes que hacen dinámica a una página Web pudiendo transformarla en lo que se conoce como *aplicación Web*. Aquí se lleva a cabo el enlace y la comunicación con la base de datos y la reactividad de ciertos elementos de la aplicación.

El JS se compone de dos partes: enlace con mongoDB y definición de *helpers* y funciones. Los *helpers* y funciones se definen especificando el *template* que los utilizará. Son privados para el resto de los *templates*.

Enlace con MongoDB

```
WaterSensors = new Mongo.Collection('sensors');
```

En la sentencia anterior se realizan dos cosas al mismo tiempo: por un lado, se crea una copia (MiniMongo) de la base de datos en el cliente llamada `sensors`, y por el otro, se crea una variable JavaScript `WaterSensors` con la que se manejará la MiniMongo creada.

Esperar al DOM (*Document Objet Model*)

Así como jQuery utiliza la función `$(document).ready()` para esperar que los objetos del HTML estén definidos para manipularlos luego, Meteor también tiene lo suyo: la función `.onRendered()`.

```
Template.morrisBar.onRendered(  function () {
  //code here
});
```

Dentro de esa función se define y, eventualmente, se llama a la función encargada de hacer el gráfico. Antes de ver el código que configura y define el gráfico es necesario presentar otra parte muy importante que hace a la reactividad del mismo frente a cambios en la base de datos: el asistente `.observe()`.

```
WaterSensors.find().observe({
  added: function () {
    drawChart()
  },
  changed: function () {
    drawChart()
  }
});
```

Básicamente, frente a *inserts* (`added:`) o *updates* (`changed:`) en la colección `WaterSensors`, se ejecutará la función `drawChart()` que se define a continuación. Notar que al momento sólo se está ordenando que se ejecute una función `drawChart()` frente a variaciones en `WaterSensors`, pero todavía no se ha definido dónde ni cómo ejecutarla.

Configuración de morris.js

Aquí se demuestra el "cómo" de la sección anterior.

```
Template.morrisBar.onRendered(function () {
  var drawChart = function () {
    $('#myfirstchart').empty();
  var data = WaterSensors.find({},{fields:{node:1,flow:1},
         sort:{node:1}}).fetch();
```

```
6
7    if (data) {
8      new Morris.Bar({
9        // ID of the element in which to draw the chart.
10     element: 'myfirstchart',
11       // Chart data records -- each entry in this array corresponds
12       // to a point on the chart.
13     data:    data,
14     // The name of the data record attribute that contains x-values.
15     xkey:    'node',
16     // A list of names of data record attributes that contain
17     // y-values.
18     ykeys:   ['flow'],
19     // Labels for the ykeys -- will be displayed when you hover
20     // over the chart.
21     labels:  ['Value'],
22     resize:  true
23     });
24     }
25   }
```

Como se ve en el código anterior, dentro de la definición de `drawChart()` se borra lo que pueda contener el elemento con id `myfirstchart`, se define `data` en la cuarta línea y se pasa a `Morris.Bar()` junto con otros elementos. Lo más interesante aquí es cómo se importan esos datos de la mongoDB y se guardan en forma de *array* en la variable `data`.

Con `.find()` se obtiene un cursor. Esto es, una especie de puntero que apunta al primer elemento del total de elementos que cumplen con el criterio de búsqueda establecido en el primer parámetro de `.find()`, que al no tener nada, trae todos los documentos de la colección `WaterSensors`. El segundo parámetro establece qué campos incluir en el resultado de la búsqueda, excluyendo los demás. El tercer y último parámetro de `.find()` nos da la opción de ordenar la lista de elementos, que en este caso, se ordenan en función del nombre del medidor de manera ascendente.

Por último, con `.fetch()` se convierte la lista de elementos apuntados por el cursor en un *array*. Debido a que los documentos en mongoDB son guardados en formato JSON, el *array* tendrá forma de JSON también.

Dónde insertar morris.js

Ya se explicó cómo definir y configurar el gráfico de barras morris.js. Además, se tuvo en cuenta refrescarlo ante cambios en la base de datos. Es momento de definir en qué lugar del `<body>` del HTML colocarlo.

El gráfico se inserta dentro del archivo `.html` encapsulado en un *template*. En Meteor, un *template* es un trozo de HTML que puede *incrustarse* en cualquier parte dentro del `<body>`. Como se ve en las líneas 13 y 14 del siguiente código, se han insertado dos *templates*: `morrisBar` y `sensorsList`.

```
<head>
  <title>WiWater</title>
</head>

<body>
  <div class="container">
    <div class="page-header">
    <h1>
     <span class="glyphicon glyphicon-tint" aria-hidden="true"></span>
     Water Sensors
    </h1>
    </div>
    {{> morrisBar}}
    {{> sensorslist}}
  </div>
</body>
```

El *template* `morrisBar` es el contenedor del gráfico de barras y se define de la siguiente manera:

```
<template name="morrisBar">
  <div id="myfirstchart" style="height: 300px; width: 450px;"></div>
</template>
```

Como se ve en el `.html` anterior, sólo se define el identificador `myfirstchart` y las dimensiones del gráfico.

El *template* `sensorsList` es una lista de los medidores dentro de la red con sus consumos totales en el día, en la semana y en el mes. Se define en el siguiente código:

```
<template name ="sensorslist">
  {{#each showSensor}}
    <ul class="list-group sensor" style="width: 400px">
      <li class="sensor list-group-item active">
        <h4>
          <span class="glyphicon glyphicon-ok" aria-hidden="true">
          </span>
          Node {{node}}
        </h4>
      </li>
      <li class="list-group-item">
        <h4><span class="label label-info">{{flow}}</span>
          litros en el dia
        </h4>
        <h4><span class="label label-warning">{{flow}}</span>
          litros en la semana
```

```
          </h4>
          <h4><span class="label label-default">{{flow}}</span>
            litros en el mes</h4>
        </li>
        <li class="list-group-item">
          actual rate
        </li>
      </ul>
    {{/each}}
```

Notar que el nombre del medidor es insertado con {{`node`}} y los valores de caudal con {{`flow`}}. Para lograr que los elementos {{`node`}} y {{`flow`}} correspondan a valores almacenados en la base de datos es necesario comunicarse con la misma y se realiza con el uso de *helpers*. Un *helper* es una función asociada exclusivamente a un *template*. A continuación se muestran los *helpers* correspondientes al *template* `sensorslist`.

```
Template.sensorslist.helpers({
    'showSensor': function () {
        return WaterSensors.find()
    },
    'countSensors': function () {
        return WaterSensors.find().count()
    },
    'selectedClass': function () {
        return "active"
    }
});
```

Ver en el código anterior que existen tres *helpers* para `sensorslist`: `showSensor`, `countSensors` y `selectedClass`. El más importante es `showSensor` que devuelve un cursor apuntando a todos los elementos de `WaterSensors` gracias a la función `.find()`.

Entonces, cuando se dispone del cursor apuntando al *array* de medidores en la base de datos, al utilizar *handlebars* {{ }} se logra *incrustar* a los valores cuyos campos se especifican dentro. Gracias a la función especial `#each` se recorre el *array* de medidores.

Importación del paquete morris.js

Para utilizar los gráficos reactivos MorrisJS [17] y poder graficar los caudales totales para cada medidor se importó al proyecto Meteor el paquete **morris.js**. El paquete morris.js se encuentra en Atmosphere [18], el repositorio oficial de paquetes Meteor. El nombre de importación del paquete es `bshamblen:morrisjs` y sus dependencias son `clubfest:raphael` y `jquery`. La dependencia `jquery` no se necesita instalar ya que es nativa de Meteor.

Definidos el `.html` y el `.js` ya se puede visualizar la aplicación Web (figura 4.11).

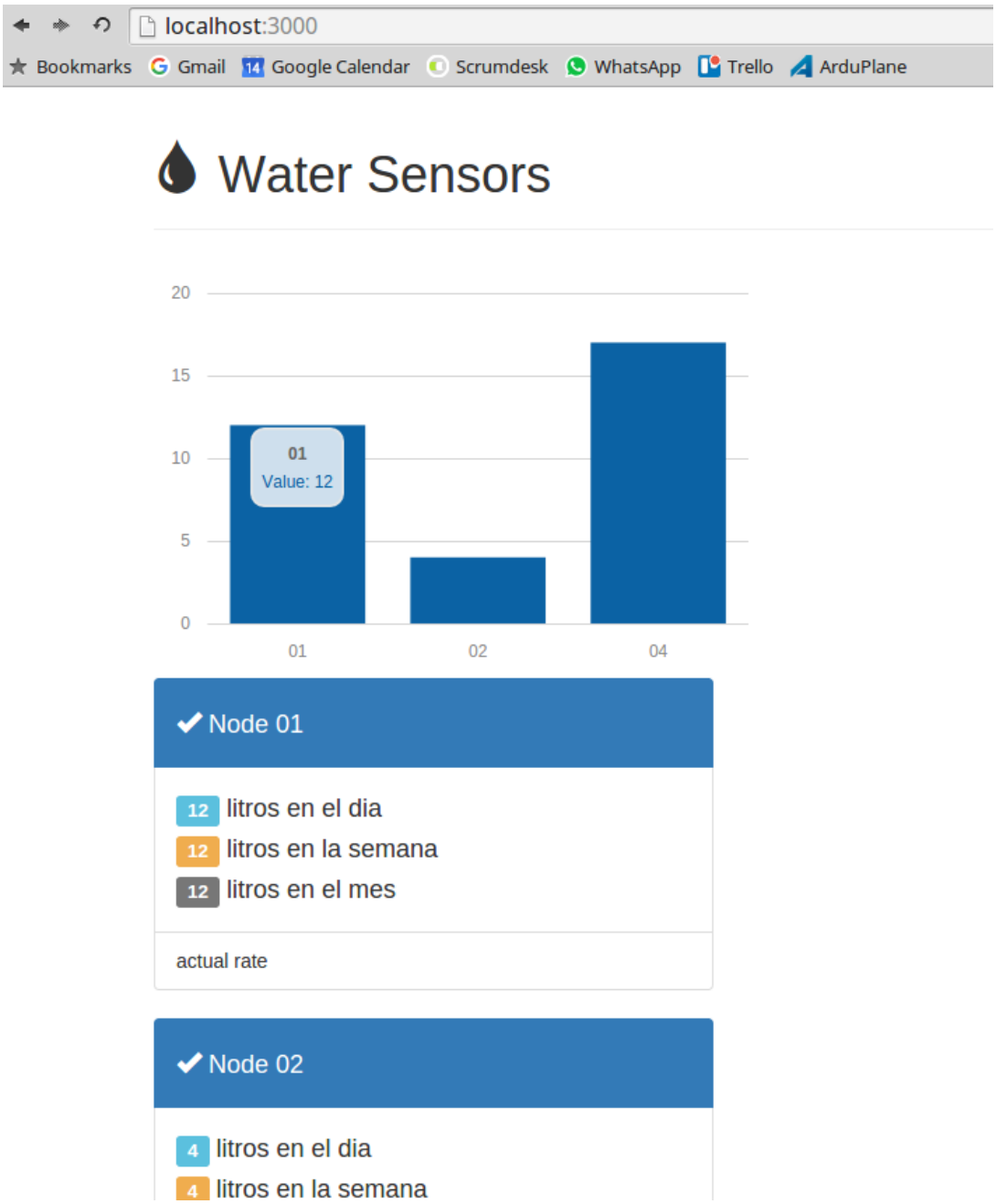


Figura 4.11: Aplicación Web multiplataforma de WiWater

Capítulo 5

Conclusiones

5.1. Resultados

Los objetivos propuestos se han alcanzado satisfactoriamente. Como se expuso en capítulos anteriores, se presentaron inconvenientes durante el diseño de algunos bloques componentes que comprometían la *performance* del sistema. Sin embargo, con actitud resiliente se pudieron superar esos obstáculos mediante un reanálisis de la situación.

En particular, los medidores autónomos han cumplido su función como tal. Su desempeño ha sido constante a lo largo de varias sesiones y en usos prolongados. Tanto la generación y transmisión de pulsos para medir caudal como el sistema de generación y acondicionamiento de potencia no se han visto afectados por usos exigentes. Sin embargo, se identificó como limitación del sistema un caudal mínimo detectable a veces impráctico frente a situaciones de consumo en el hogar. Como ya se ha mencionado, caudales menores a 2.2 L/min no podrán ser medidos por el sistema. Aunque este caudal no es común en la mayoría de las actividades de consumo hogareño (figura 3.5), disminuir su valor sería un requisito para mejorar el sistema/producto y poder prometer, como diseñador, la detección de caudales aún más bajos, incluyendo pérdidas en las cañerías.

Con respecto al concentrador de datos, se ha hecho un gran esfuerzo por implementar

lo más novedoso en cuanto a prestaciones, innovación y características para que el sistema desarrollado sea fácilmente incorporado por las generaciones presentes y futuras. Hoy en día es determinante que un producto se adapte a las costumbres de los usuarios. El producto debe adaptarse al comportamiento y hábitos del usuario, y no al revés. Aunque optar por una aplicación Web de tiempo real para mostrar las mediciones ha sido acertado, el esfuerzo asociado al aprendizaje e implementación de nuevas tecnologías (algunas en fase de prueba o *beta*) ha sido importante.

El sistema completo puede verse funcionando en un depósito de agua tipo mochila para inodoro en el siguiente *link* de YouTube: `https://youtu.be/gW4hDlnjXsA`

5.2. Limitaciones y mejoras

Se puede asegurar, como con casi cualquier desarrollo de base tecnológica, que el sistema presentado puede ser aún mejor. Su uso constante en el mundo real arrojaría información importante para la ejecución de mejoras, tanto del lado de los medidores como del lado de la interfaz con el usuario y los servicios ofrecidos.

Como se mencionó en la sección anterior, una disminución del caudal mínimo detectable mejoraría considerablemente el desempeño del sistema y le daría una mejor posición en el mercado en el caso de una eventual comercialización. Para lograr este objetivo sería necesario remplazar el generador adquirido de un tercero por uno diseñado específicamente para caudales más bajos. Se puede usar como base en el diseño el generador presentado en la *alternativa B* de la sección 3.1.3. De hecho, se continúa investigando e invirtiendo recursos con el fin de mejorar el sistema propuesto para colocarlo en el mercado. La adquisición de una impresora 3D (figura 5.1) permitirá construir mediante fabricación positiva un prototipo de generador hidroeléctrico acorde a los caudales típicos en un hogar. La fabricación rápida de prototipos junto a la simulación previa del mismo mediante *software* de simulación multifísica como *Comsol Multiphysics* (figura 5.2) permitirán un acelerado proceso de mejora constante y en consecuencia una rápida colocación en el mercado.

Es interesante pensar que avances relacionados solamente a este sistema podrían revelar nuevas técnicas asociadas a la extracción de energía, formas de implementar electrónica de ultra-bajo consumo, optimización de comunicaciones y redes inalámbricas ultra-bajo consumo, mejora de interfaz para el *Internet de las Cosas* y una gran cantidad de avances que en la actualidad permanecen ocultos.

5.3. Metodología de trabajo empleada

Durante el desarrollo de este trabajo final se ha empleado la metodología ágil de trabajo *Scrum* para definir, gestionar y alcanzar los distintos objetivos que exigió la concreción del sistema/producto.

Esta metodología es particularmente útil cuando se aplica a proyectos que poseen matices de investigación (como es el caso de este trabajo) ya que, si bien el objetivo puede ser claro, el camino hacia el mismo puede variar constantemente. Este método de trabajo se adapta perfectamente a este tipo de escenario cambiante, propenso a situaciones

Figura 5.1: Impresora 3D M3D [19].

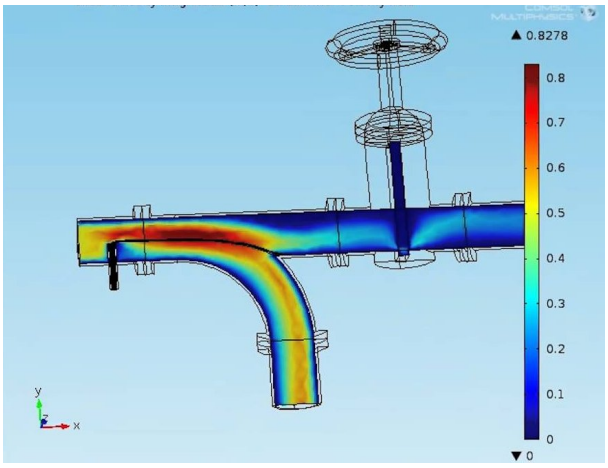


Figura 5.2: Simulación multifísica con Comsol [20].

imprevistas (y no por ello consideradas siempre negativas), en donde los requerimientos se realimentan con los resultados y una larga lista de factores, conocidos o no, pueden destruir por completo una rígida planificación basada en metodologías más tradicionales.

Se utilizó una herramienta digital llamada *ScrumDesk* [21] como complemento para llevar a cabo la gestión ágil de todo el proyecto.

Apéndice A

Macros de SimpliciTI

Macro	Default value	Comments
MAX_HOPS	3	Frames replayed only 3 times maximum
MAX_HOPS_FROM_AP	1	Maximum distance from an AP. Used by poll and poll response to limit network replay traffic.
MAX_APP_PAYLOAD	10	Absolute maximum is based on size of Rx and Tx FIFOs (50 for CC2500/CC1100 class including SoCs and 111 for IEEE radios CC2430/CC2520).
DEFAULT_JOIN_TOKEN	0x01020304	Change this to provide some access security
DEFAULT_LINK_TOKEN	0x05060708	Should be changed by APs on AP networks. Customer should modify **nwk_join.c:generateLinkToken()**
FREQUENCY_AGILITY	Not defined	When defined enabled support for Frequency Agility. Otherwise only the first entry in the channel table is used.
APP_AUTO_ACK	Not defined	Adds support for application level auto acknowledgment. Requires Extended API
EXTENDED_API	Not defined	Enables **SMPL_Unlink()**, **SMPL_Ping()** and **SMPL_Commission()**.
NVOBJECT_SUPPORT	Not defined	Adds support for gerting and setting connection context for saving across resets.
SMPL_SECURITY	Not defined	Enables security infrastructure

Figura A.1: Macros de configuración de red.

Macro	Default value	Comments
NUM_CONNECTIONS	4	Number of connections supported. Each connection supports bi-directional communication. Access Points and Range Extenders can set this to 0 if they do not host End Devices.
SIZE_INFRAME_Q	2	Two is probably enough for an End Device. Little bit larger for REs and APs. AP needs larger input frame queue if it is supporting store-and-forward clients because the forwarded messages are held here.
SIZE_OUTFRAME_Q	2	The output frame queue can be small since Tx is done synchronously. If an Access Point device is also hosting an End Device that sends to a sleeping peer the output queue should be larger because the waiting frames in this case are held here. Actually 1 is probably enough otherwise.
THIS_DEVICE_ADDRESS	{0x78,0x56,0x34,0x12}	This device's address. The first byte is used as a filter on the CC1100/CC2500 radios so THE FIRST BYTE MUST NOT BE either 0x00 or 0xFF. Also, for these radios on End Devices the first byte should be the least significant byte so the filtering is maximally effective. Otherwise the frame has to be processed by the MCU before it is recognized as not intended for the device. APs and REs run in promiscuous mode so the filtering is not done. This macro initializes a static const array of unsigned characters of length **NET_ADDR_SIZE** (found in nwk_types.h).
Range Extender		
RANGE_EXTENDER		Device type declaration for Range Extenders
End Device		
END_DEVICE		Device type declaration for End Devices
RX_POLLS	Not defined	Define RX_POLLS of the device is a polling End Device.
Access Point		
ACCESS_POINT		Device type declaration for Access Points
NUM_STORE_AND_FWD_CLIENTS	3	Number of store-and-forward clients supported.
AP_IS_DATA_HUB	Not defined	If this macro is defined the AP will be notified through the callback each time a device joins. The AP should be running an application that listens for a link message on receipt of this notification. The ED joining must link immediately after it receives the Join reply.

Figura A.2: Macros de configuración de dispositivo.

Índice de figuras

1.1. *WiWater* en un hogar convencional. 6

2.1. Medidores instalados en red de agua dentro del hogar. 8
2.2. Sistema hidráulico a modelizar. 10
2.3. Sistema hidráulico con variables eléctricas. 11
2.4. Modelo eléctrico equivalente. 11
2.5. Red de sensores y concentrador en topología estrella. 12
2.6. Esquema generalizado del medidor. 13
2.7. Esquema generalizado del concentrador de datos. 14
2.8. Medidores en el mercado, con batería y sin *wireless link*. 14
2.9. Método *Device.Block.Stage*. 15

3.1. Diagrama de bloques del *hardware* del medidor. 18
3.2. Diagrama de bloques del generador. 19
3.3. Generadores adquiridos en *Aliexpress*. 20
3.4. Corte del modelo 3D del generador. 21
3.5. Función de transferencia V_{out} vs. *Caudal*. 22
3.6. Comportamiento del sensor de efecto Hall. 24
3.7. Esquemático en *datasheet*. 25
3.8. Esquemático en *software* KiCad 26
3.9. Sensor de efecto Hall en encapsulado SOT23. 26
3.10. Modelo 3D de sensor instalado en generador. 26
3.11. Fotografía del sensor Hall instalado en la cámara del rotor húmedo. . . . 27
3.12. Prueba de sensibilidad y respuesta del sensor de efecto Hall. 27
3.13. Función de transferencia simplificada y dividida en etapas. 28
3.14. Alternativa A: *Buck-Boost* + Capacitor + Medidor en serie. 29
3.15. Alternativa A. Etapa 1. 30
3.16. Alternativa A. Etapa 2. 31
3.17. Capacitores de almacenamiento *EnerChip*. 32
3.18. Medidor de caudal por pulsos. 32
3.19. Alternativa B: Generador teórico de baja inercia + *Buck-Boost*. 33
3.20. Prototipo de generador AFPM (*Axial Flux Permanent Magnet*.) 34
3.21. Alternativa B. Etapa 1. 34
3.22. Alternativa B. Etapa 2. 35
3.23. Alternativa C: Regulador lineal de bajo *drop-out* + Filtro. 36
3.24. Alternativa C. Etapa 1. 37
3.25. Alternativa C. Etapa 2. 38
3.26. TLV70133 al lado de pinza para SMD. 39
3.27. Diseño y fabricación del *PCB* de acondicionamiento de potencia. 41

3.28. Arriba: voltaje sin filtrar y sin regular. Abajo: voltaje filtrado y regulado. 42
3.29. Regulador lineal de bajo *drop-out* y capacitores de filtrado. 42
3.30. *MSP430F2x MCUs Key Features* [8]. 43
3.31. Bloques funcionales del microcontrolador MSP430F2274 [9]. 44
3.32. eZ430-RF2500T: MSP430F2274 + CC2500. 44
3.33. Esquemático de conexión entre MSP430F2274 y CC2500 [10]. 45
3.34. eZ430-RF2500T ensamblado a PCB de acondicionamiento. 46
3.35. Arriba: salida del sensor Hall. Abajo: alimentación del sistema. 47
3.36. Diagrama de bloques del *transceiver* CC2500 [11]. 49
3.37. Esquemático CC2500 + Adaptación + Antena [11]. 50
3.38. BOM del CC2500 con adaptación de antena. 50
3.39. Diagrama de flujo del *firmware* del medidor. 53
3.40. Capas del *firmware*. 54
3.41. SimpliciTI: sesión ejemplo [13]. 56

4.1. Diagrama de bloques del *hardware* del concentrador. 61
4.2. Interfaz USB para programación/*debugging* y UART. 62
4.3. eZ430-RF2500 con interfaz UART-USB eZ430-RF USB. 62
4.4. Mini-computadora Raspberry Pi 2 Model B. 64
4.5. Concentrador de datos. 64
4.6. Diagrama de flujo del *firmware* del concentrador. 66
4.7. *Screenshot* de Raspbian. 68
4.8. Paquetes recibidos y procesados por Python 71
4.9. Plataforma Meteor y recursos [15]. 76
4.10. Subconjunto de la base de datos en el cliente [15]. 77
4.11. Aplicación Web multiplataforma de WiWater 82

5.1. Impresora 3D M3D [19]. 85
5.2. Simulación multifísica con Comsol [20]. 85

A.1. Macros de configuración de red. 86
A.2. Macros de configuración de dispositivo. 87

Listado de Acrónimos

IP — *Internet Protocol*
HTML — *HyperText Markup Language*
JS — *JavaScript* BOM — *Bill Of Materials*
PCB — *Printed Circuit Board*
FET — *Field-Effect Transistor*
CAD — *Computer-Aided Design*
AFPM — *Axial Flux Permanent Magnet*
MCU — *Micro-Controller Unit*
LED — *Light-Emitting Diode*
RF — *Radio Frequency*
ISM — *Industrial, Science and Medical*
SRD — *Short Range Devices*
FIFO — *First Input First Output*
SPI — *Serial Protocol Interface*
IF — *Intermediate Frequency*
ADC — *Analog to Digital Converter*
AGC — *Automatic Gain Control*
BSP — *Board Support Package*
HAL — *Hardware Abstraction Layer*
API — *Application Programming Interface*
LBT — *Listen Before Talk*
DCO — *Digitally Controlled Oscillator*
DC — *Direct Current*
LPM — *Low Power Mode*
CPU — *Central Processing Unit*
UART — *Universal Asynchronous Receiver-Transmitter*
RAM — *Random Access Memory*
HDMI — *High Definition Media Interface*
GPIO — *General Pourpose I/O*
WAF — *Web Application Framework*
DDP — *Distributed Data Protocol*
DOM — *Document Object Model*

Bibliografía

[1] Papa Francisco. *Encíclica Laudato si': Sobre el cuidado de la casa común.* Opus Dei Oficina de Información, 2015.

[2] Robert Jerome Glennon. *Unquenchable: America's water crisis and what to do about it.* Island Press, 2009.

[3] Thomas Boyle, Damien Giurco, Pierre Mukheibir, Ariane Liu, Candice Moy, Stuart White, and Rodney Stewart. Intelligent metering for urban water: A review. *Water*, 5(3):1052–1081, 2013.

[4] Katsuhiko Ogata. *Modern Control Engineering (5th Edition).* Prentice Hall, 2009.

[5] Aliexpress. Comercio electrónico. *http://www.aliexpress.com*, 2015.

[6] Cymbet Corporation. Enerchip smart solid state batteries. *http://www.cymbet.com/products/enerchip-solid-state-batteries.php*, 2015.

[7] Wang Song Hao and Ronald Garcia. Development of a digital and battery-free smart flowmeter. *Energies*, 7(6):3695–3709, 2014.

[8] Texas Instruments. Msp430f2x mcus key features. *http://www.ti.com/lsds/ti/microcontrollers*, 2015.

[9] Texas Instruments. Msp430f22x2, msp430f22x4 mixed signal microcontroller datasheet. *http://www.ti.com/lit/ds/symlink/msp430f2274.pdf*, 2015.

[10] Texas Instruments. ez430-rf2500 development tool. *http://www.ti.com/lit/ug/slau227f/slau227f.pdf*, 2015.

[11] Texas Instruments. Low-cost low-power 2.4 ghz rf transceiver. *http://www.ti.com/lit/ds/symlink/cc2500.pdf*, 2015.

[12] Texas Instruments. Simpliciti api. *http://www.ti.com/corp/docs/landing/simpliciTI/*, 2008.

[13] Texas Instruments. Simpliciti overview. *http://www.ti.com/lit/ml/swru130b/swru130b.pdf*, 2015.

[14] MongoDB. Pymongo API. Oficial python driver for MongoDB. *https://api.mongodb.org/python/current/api/pymongo/index.html*, 2015.

[15] Meteor. Meteor, the javascript app platform. *https://www.meteor.com/*, 2015.

[16] Meteor Universal Blog. 1st time installation: meteor universal on raspbian wheezy. *http://meteor-universal.tumblr.com/post/118809833179/1st-time-installation-meteor-universal-raspbian-wheezy*, 2015.

[17] Olly Smith MorrisJS. Good-looking charts. *http://morrisjs.github.io/morris.js/*, 2015.

[18] Percolate Studio. Atmosphere, oficial meteor packages. *https://atmospherejs.com/*, 2015.

[19] M3D. The m3d printer. *https://printm3d.com/themicro/*, 2015.

[20] COMSOL Inc. Comsol multiphisics. *https://www.comsol.com/*, 2015.

[21] ScrumDesk. Agile management. *http://www.scrumdesk.com/*, 2015.